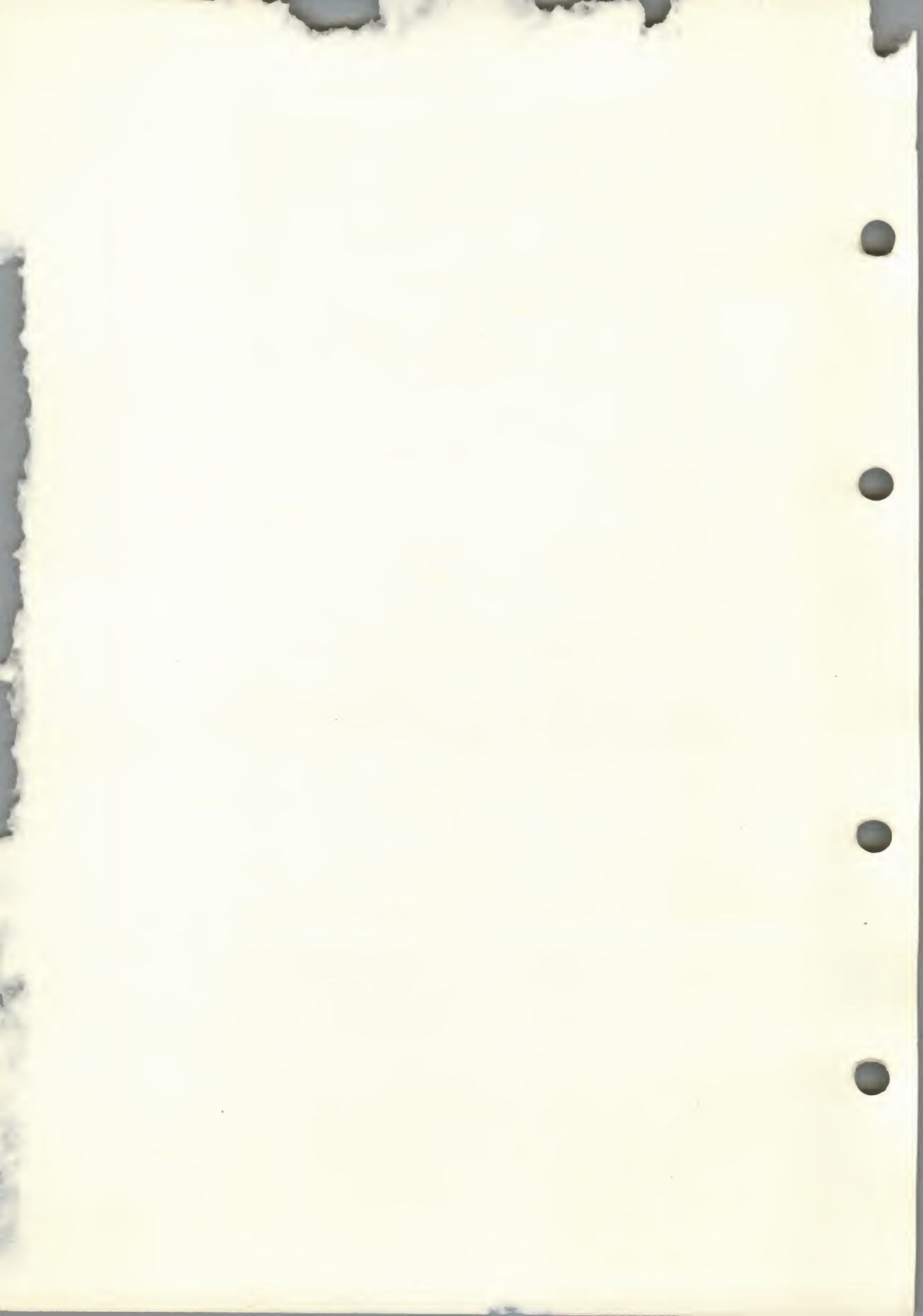




HANS VAN KAMPEN
RIEN DE KUIPER



VOORWOORD

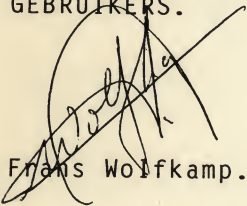
Gaarne wil ik iedereen bedanken die mee heeft geholpen bij het tot stand komen van dit boek.

Hierbij denk ik vooral aan Cindy, mijn vrienden, kennissen en niet te vergeten de beide auteurs van Kampen en de Kuiper.

Verder wil ik de volgende detaillisten bedanken voor hun medewerking t.w.:

Ir. bureau Schroder Eindhoven
Vlasveld Electronica Leiden
Capi Lux Amsterdam.

Ik hoop dat dit boek met zijn toekomstige updates als een onmisbare schakel mag fungeren tussen de APPLE en haar GEBRUIKERS.



Frans Wolfkamp.

Ten Geleide :

Voor U ligt het eerste Supplement van het APPLE HANDBOEK. Behalve een zeer uitgebreide behandeling van het Disk Operating System vinden we nog de volgende onderwerpen :

1. Een globale kennismaking met twee APPLE compatibele Micro-Computer Systemen, n.l. de PEARCOM en de BASIS 108. Deze Systemen komen respectievelijk uit Engeland en Duitsland.

2. Een korte verkenning van de mogelijkheden voor APPLE in het komende jaar in het licht van de concurrentie van de onder 1. genoemde Systemen.

3. Een beschouwing over de Softcard van Microsoft. Dit is een Interface met een Z80 Microprocessor die de APPLE in staat stelt onder CP/M te kunnen draaien. Met de introductie kwam er een dimensie voor de APPLE II bij. De mogelijkheden hiervan zijn zo groot, dat hierover een afzonderlijk Supplement gepubliceerd zal worden.

De auteurs hebben veel reacties op het APPLE HANDBOEK gehad. De overwegend opbouwende kritiek en de talrijke suggesties hebben ons gesteund in onze overtuiging dat dit HANDBOEK een wezenlijke functie vervult. Wij willen vanaf deze plaats een ieder die reageerde heel hartelijk danken en spreken de hoop uit dat dit zich in de toekomst zal voortzetten.

Gorinchem, April 1982

Hans van Kampen
Rien de Kuiper



HANS VAN KAMPEN
RIEN DE KUIPER



1954
1955

omslagtekening: Wichert van Engelen
drukkerij : Total Photo Amsterdam

ISBN 90-70556-01-4

Copyright©1981 1e.druk

Uitgeverij Wolfkamp
Amsterdam

Niets uit deze uitgave mag worden verveelvoudigd en/of openbaargemaakt door middel van druk, fotokopie, microfilm of op welke andere wijze, zonder voorafgaande schriftelijke toestemming van de uitgever.

INLEIDING

Zonder twijfel zal het APPLE II Personal Computer Systeem uw gedachten omtrent computers veranderen.

De APPLE II is immers opgewassen tegen de meest uiteenlopende taken. Het is een weldoordachte computer, die robuust is en die mogelijkheden in zich verenigt, die elke fantasie tarten.

De APPLE II is op dit moment het meest consistente microcomputer concept met een bijna onvoorstelbare uitbreidingsmogelijkheid. Het marktaandeel, dat APPLE COMPUTERS INC. heeft opgebouwd, duidt erop, dat de grote betrouwbaarheid van haar systemen en het geslaagde ontwerp ervan weerklank heeft gevonden over de gehele wereld.

Dit nieuwe APPLE II HANDBOEK is principieel bestemd voor elke APPLE bezitter, maar is ook geschikt voor degenen, die zich nog moet oriënteren op het gebied van micro-automatisering. Dit APPLE II HANDBOEK behandelt het gebruik van deze microcomputer en voert de lezer/gebruiker systematisch door het systeem, dat natuurlijk meer omvat dan de fysieke microprocessor.

Toch is dit HANDBOEK geen lesboek. De behandelde onderwerpen worden op een luchtige wijze gebracht en nodigen uit tot zelfwerkzaamheid. Wel wordt verondersteld, dat de lezer de onderwerpen zal begrijpen, alvorens over te gaan op aansluitende maar meer gedetailleerde besprekingen.

Het gedeelte, dat de systeemhardware omvat, is eenvoudig van opzet, zodat ook niet-electronici hierin hun weg kunnen vinden. De bespreking van het assembleren, alsmede het DOS is praktisch van opzet, zonder dat te ingewikkeld wordt gedaan.

Dit APPLE II HANDBOEK is een onmisbare schakel tussen gebruiker en microcomputersysteem. Het losbladige boek biedt de mogelijkheid van tussentijdse aanpassing en uitbreiding.

Zowel de beginner als de gevorderde gebruiker zal dit boek weten te waarderen.

DE AUTEURS

Rien de Kuiper en Hans van Kampen zijn beide APPLE-enthousiasten, die een meerjaren ervaring hebben met microcomputers. Rien werd op 10 mei 1957 geboren en kwam na de Middelbare School in de computerwereld terecht. Om het grote 'mainframe'-gebeuren beter te doorgronden, kocht hij een APPLE computer. Hij is specialist op het gebied van tekstverwerking en vormgeving.

Hans van Kampen is geen onbekende in de microcomputerwereld. In 1978 stelde hij het eerste Apple Handboek samen, dat meer was toegesneden op de Amerikaanse standaard-APPLE. Afgezien van dit boek schreef en vertaalde hij talrijke boeken over luchtvaart en controversiele onderwerpen. Ook publiceerde hij artikelen over automatisering in diverse periodieken (onder pseudoniem).

Beide auteurs houden zich graag aanbevolen voor opbouwende kritiek en suggesties. Tevens zullen zij geïnteresseerden verder willen wijzen in het gebeuren rond de APPLE-computer. Wie met de auteurs in contact wil treden, richt zich tot Postbus 2030, 4200 BA Gorinchem/Nederland. of tot de uitgever: postbus 70254 1007 KG Amsterdam/Nederland

De Auteurs worden gaarne op de hoogte gehouden van nieuwe APPLE programmatuur, welke in Nederland en België verschijnt.

HOOFDSTUK 1 Wegwijzer

Ten gerieve van de lezer werd in dit APPLE II HANDBOEK een uitgebreide index opgenomen. Hierdoor is het mogelijk snel een gezocht trefwoord of commando-statement terug te vinden.

Bovendien verschaft de nu volgende inhoudsopgave een duidelijk beeld van de opbouw van het boek.

Door index, inhoudsopgave en commando-wijzer voorin te plaatsen, kan de gebruiker op een efficiënte wijze toegang krijgen tot de leesstof.

De bladzijden zijn eenduidig gemerkt:

Eerst volgt het hoofdstuknummer, daarna de bladzijde binnen het hoofdstuk.

Dus: Systeem-commando's 5.1 betekent, dat in Hoofdstuk 5 Blad 1 het onderwerp "systeem commando's" een aanvang neemt.

De commando-wijzer is een index betreffende de Applesoft-instructies. De statements zijn gegroepeerd naar hun aard.

1. Inhoudsopgave

HOOFDSTUK 2 Hoe te beginnen

1. Het systeem opstarten	2.1
2. Iets over commando's	2.4
3. Programma's corrigeren	2.8
4. Wat is Applesoft ?	2.16

HOOFDSTUK 3 Programmeren in Basic

1. Het getallensysteem	3.1
2. Het print-formaat	3.4
3. Variabelen benoemen	3.6
4. Logische & rekenkundige regels	3.11
5. Nog iets over commando's	3.13

HOOFDSTUK 4 Enkele overzichten

1. Commando samenvatting	4.1
2. Iets over Integer Basic	4.3
3. ASCII-codes	4.7
4. Floating point package	4.8

HOOFDSTUK 5 Beschikbare commando's

1. Systeem commando's	5.1
2. Editing commando's	5.8
3. Over array's & strings	5.13
4. Input & output commando's	5.19
5. Flow control commando's	5.27
6. Spel-, of game control	5.28

HOOFDSTUK 6 Ingebouwde functies

1. Wiskundige functies	6.1
2. PEEK, POKES & CALL's	6.6
3. Automatische foutmeldingen	6.8
4. Gereserveerde woorden	6.11
5. Applesoft geheugenmap	6.13

HOOFDSTUK 7 Grafische voorstellingen

1. Grafieken met grof raster	7.1
2. Grafieken met fijn raster	7.6
3. Het maken van figuren	7.10

HOOFDSTUK 8 Monitor

1. Enkele observaties	8.1
2. De monitor in en uit	8.3
3. Geheugen commando's	8.7
4. Display commando's	8.8
5. Mini-assembler	8.12
6. Pseudo machine interpreter	8.37

HOOFDSTUK 9 Disk drives

1. Het gebruik van disk drives	9.1
2. Tracks & sectoren	9.2
3. Het hanteren van de diskette	9.4
4. Welk disksysteem	9.5
5. Van pr# tot catalog	9.6
6. Directe DOS commando's	9.8
7. DOS programma commando's	9.11
8. DOS lees / schrijf operaties	9.23

HOOFDSTUK 10 Interfacing & I/O

1. Overzicht	10.1
2. Video	10.2
3. Voeding	10.4
4. Cassette recorder	10.5
5. Keyboard	10.7
6. Game I/O	10.8
7. Geluid	10.9
8. Slots	10.10
9. Nadere beschouwing	10.13

HOOFDSTUK 11 Hardware

1. Synchronisatie & timing	11.1
2. ROM & PROM beknopt	11.15
3. RAM beknopt	11.15
4. De Geheugenmap beknopt	11.15

HOOFDSTUK 12 Trouble Shooting

1. Enkele oorzaken	12.1
2. Vervangen van IC's	12.3

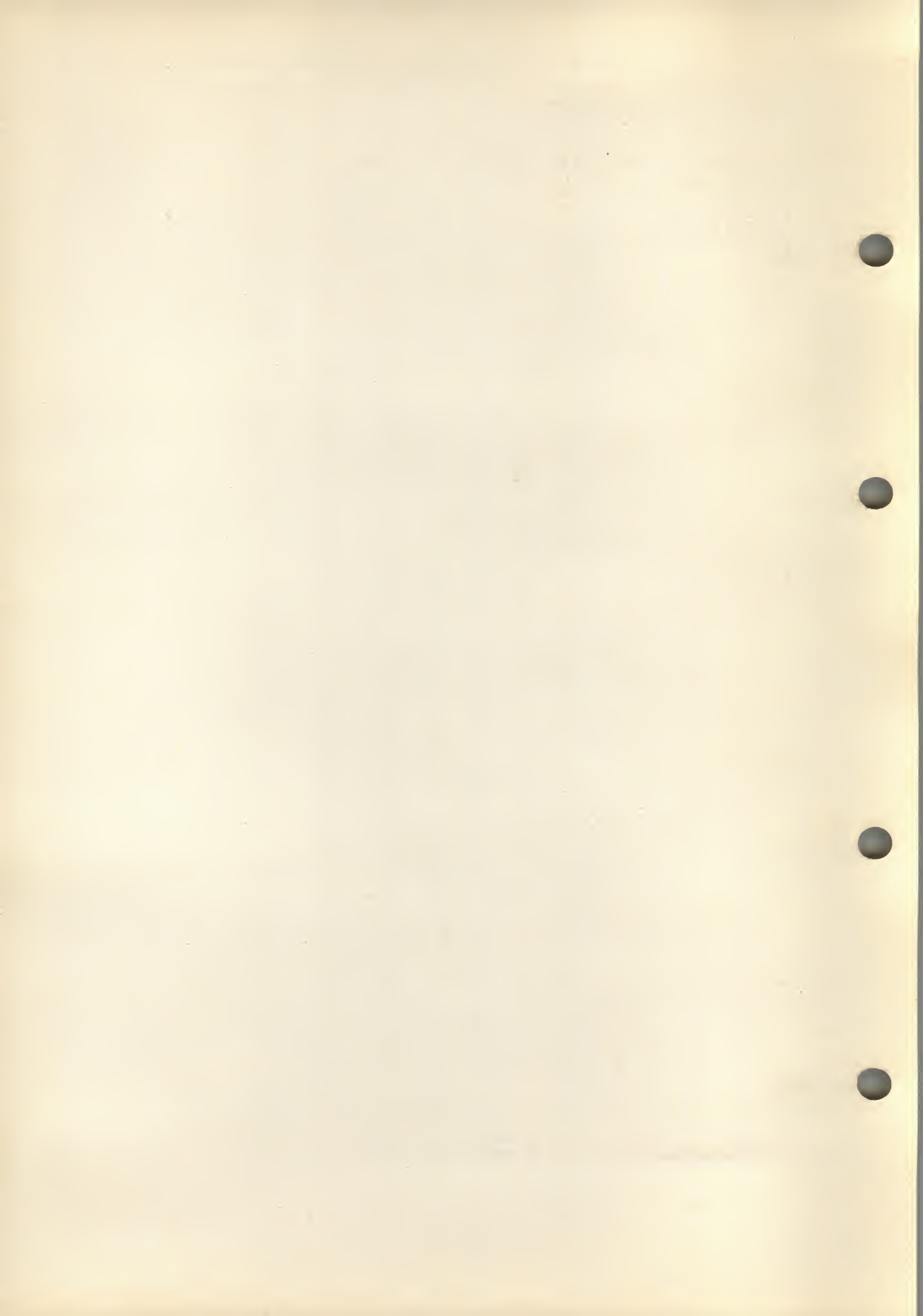
HOOFDSTUK 13 Nabeschouwing

1. Hobby Computer Club	13.1
2. Conflict Simulatie (Ducosim)	13.2
3. Daar maken we een copietje van	13.4
4. Ten Besluite	

H00FDSTUK 14 algemeen

1. Evaluatie voorjaar 1982	14.1
2. Pearcom	14.2
3. Z-80A Softkaart	14.4
4. Basis 108	14.6
5. M-Basic 80	14.7

-



3. Commando-wijzer

3.1 Systeem commando's

LOAD	5.1
SAVE	5.1
NEW	5.1
RUN	5.2
END	5.2
CONT	5.3
TRACE	5.3
NOTRACE	5.3
PEEK	5.4
POKE	5.4
WAIT	5.4
CALL	5.5
HIMEM:	5.5
LOMEM:	5.6
USR	5.7

3.2 Editing commando's

LIST	5.8
DEL	5.8
REM	5.9
VTAB	5.9
HTAB	5.9
TAB	5.10
POS	5.10
SPC	5.10
HOME	5.11
CLEAR	5.11
FRE(0)	5.11
FLASH	5.12
INVERSE	5.12
NORMAL	5.12
SPEED=	5.12

3.3 Array's & Strings

DIM	5.13
LEN	5.13
STR\$	5.13
VAL	5.15
CHR\$	5.16
ASC	5.16
LEFT\$	5.16
RIGHT\$	5.17
MID\$	5.17
STORE	5.17
RECALL	5.18

3.4 Input/Output

INPUT	5.19
GET	5.22
DATA	5.23
READ	5.23
RESTORE	5.24
PRINT	5.24
IN #	5.25
PR #	5.25
LET	5.25
DEF	5.26
FN	5.26

3.5 Flow control

GOTO	5.27
IF..THEN	5.27
FOR..NEXT	5.28
GOSUB	5.29
RETURN	5.29
POP	5.30
ON..GOSUB	5.30
ON..GOTO	5.30
ONERR..GOTO	5.30
RESUME	5.31

3.6 Game control

PDL	5.32
-----	------

3.7 Wiskundige functies

SIN	6.1
COS	6.1
TAN	6.1
ATN	6.1
INT	6.2
RND	6.2
SGN	6.2
ABS	6.2
SQR	6.3
EXP	6.3
LOG	6.3
Afgeleiden	6.3 ev.

3.8 Grafieken

GR	7.3
TEXT	7.3
COLOR	7.4
PLOT	7.4
HLIN	7.5
VLIN	7.5
SCRN	7.5
HGR	7.6
HGR2	7.6
HCOLOR	7.7
HPlot	7.7
DRAW	7.15
XDRAW	7.15
ROT	7.15
SCALE	7.16
SHLOAD	7.16

3.9 D.O.S. commando's

PR#(slot)	9.6
CATALOG	9.6
LOAD	9.8
RUN	9.8
INIT	9.8
SAVE	9.9
LOCK	9.10
UNLOCK	9.10
RENAME	9.10
DELETE	9.10
VERIFY	9.10
CHR\$(4)	9.11
OPEN	9.12
WRITE	9.12
CLOSE	9.12
READ	9.13
APPEND	9.14
POSITION	9.14
BYTE	9.16
EXEC	9.18
BRUN (1)	9.19
MAXFILES	9.20
MON	9.20
NOMON	9.20
TRACE	9.21
BSAVE	9.21
BLOAD	9.21
BRUN (2)	9.22
CHAIN	9.22

TREFWOORDENREGISTER

A

ASCII-codes 4-7, 8-5
AUTO 4-4
Aan- en uitzetten 2-6, 2-7
Aanhalingstekens 2-6
Accessoires 5-31, 7-1
Accumulator 5-7, 8-10
Adres 8-1
Adressering 8-10
Alfabet 8-11
Alfanumeriek 3-6
Ampersand-teken (&) 6-11
Applesoft 2-4, 2-16, 6-12
Arithmetische waarden 5-21
Array 5-13
Assembler 2-17, 8-12 ev.
Assembly language 2-16
Autostart ROM 2-2, 2-11, 8-1

B

BASIC 2-18
Backup copie 9-8, 13-4
Bedrijfszekerheid 10-4
Beeldblokjes 7-3
Beschermen 5-6, 8-5
Beschrijving 8-10
Bit 8-4
Boolese vergelijking 3-12
Bootstrap 8-1
Buffers 9-20
Bus 11-1 ev.
Byte 8-4

C

CALL-151 2-17, 8-1 ev.
CLEAR 3-7
CTRL-B 2-13, 4-3
CTRL-C 2-13, 2-19, 5-22, 8-3
CTRL-D 9-11
CTRL-RESET 2-13
CTRL-S 5-8
CTRL-X 2-14
CTRL-functies 2-14 ev.
CTRL-toets 2-8 ev.
Caret (^) 8-12
Cassetteband 5-17, 8-13, 10-5
Cassetterecorder 5-1, 7-14, 10-5
Clobbering 12-2
Commando's 5-1 ev., 6-12
Communiceren 5-19
Compatibel 9-5
Compiler 2-17
Concatenatie 5-15, 6-9
Conflict Simulatie 13-2
Control, tekst 6-6
Coördinaat 7-4
Copie 13-4
Cursor 2-2, 2-9, 5-9

D

DEL 2-14
DMA-controller 11-2
DOS 6-12, 9-1 ev.
DSP 4-4
Darlington circuit 10-9
Data-lijst 5-13, 5-23
Debug mode 5-3
Deferred mode 3-13
Density 9-16
Dimensioneren 5-13
Disk Interface Kaart 9-6
Disk stations 9-1 ev., 12-2
Diskettes 9-4
Display controls 6-6
Ducosim 13-2
Dummy 5-26

E

E-teken 3-2
END 2-6, 4-5
ESC functies 2-14 ev.
ESC-toets 2-2, 2-8
Edit mode 2-8
Editing 2-8, 2-10, 2-12
Elementen 5-23
Encoder board 2-13
Energieverbruik 2-7
Executie 3-13, 5-6, 9-18
Ezelsbrug 3-11

F

Figuren 7-10
File 9-11 ev.
Flag 5-30, 9-13
Floating point 3-1
Floating point routines 4-8
Floppy disk 9-4
Formatteren 9-9
Fout 6-8
Foutcode 5-31, 6-7
Foutmelding(en) 4-6, 5-30, 6-8
Frequentiemodulatie 9-8
Functie 5-26, 6-1

G

Game paddles 8-13
Geheugenblok 8-5
Geheugenmap 6-13 ev., 8-4, 11-15
Geheugenplaats 5-14
Geheugenruimte 3-9
Geheugenwoord 8-1
Gereserveerde woorden 2-11, 3-6, 4-2, 6-11
Getallen 3-1
Grafics 7-1 ev.

H

HGR2 programma 7-7
HIMEM: 2-14, 7-14
Hangt, Systeem 2-19, 9-9, 12-2
Hardware 10-1
Hexadecimale waarde 7-12
Hexadecimale notatie 8-4
Hobby Computer Club 13-1

Handboeken 12-2

I

IC's 12-3

INPUT 4-5, 5-19

Immediate mode 3-13

Indirecte commando's 2-5

Initialiseren 9-8

Integer 3-1, 3-8

Integer BASIC 2-17, 4-3, 6-12

Integer variabelen 3-8

Interpreter 2-17, 4-3

J

K

Kaart, PAL 2-1, 7-1, 10-13

Kaart, RGB 2-1, 7-1, 10-2

Klachten 12-1

Koelen 10-13

Komma 2-4, 3-1

Konditionele pauze 5-4

Kopieren 13-4

L

LEN 3-9

LIST 2-6

LSB 8-4

Listing 2-7

Literal 3-8, 5-21

Loop 5-13, 5-28

Lussen 5-28

M

MAN 4-4

MOD 4-4, 6-5

MSB 8-4

Macht-teken (^) 3-1

Mantisse 3-2

Master 9-6

Mathematische functies 6-1 ev.

Megabyte 9-5

Memtop 5-4

Mini-assembler 8-12

Mini-assembler instructies 8-14 ev.

Mnemonics 2-16

Modulator 10-2

Moederbord 10-1

Monitor 2-13, 8-1 ev.

N

NEW 2-7

Nulstring 3-9, 5-23

O

Opbergdozen 9-4

Opcode 8-10 ev.

Operator 3-11

Opstarten 2-1

P

POP 5-29

PRINT 2-4, 3-4

PROM 2-16

Periferal 5-25, 10-4

Pijltoetsen (→/←) 2-9

Plotten 7-6

Plotvector 7-10

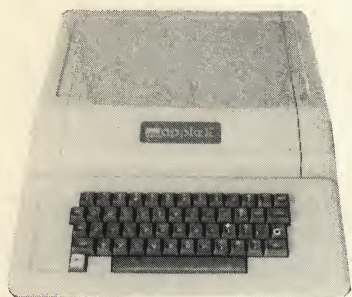
Plotvenster 7-3
 Pointer 5-23
 Prompt 2-2, 2-17
 Q
 QWERTY 10-7
 R
 RAM-blok 11-2
 REPT toets 2-9, 10-7
 RESET 2-13, 5-25
 ROM, Autostart 2-2
 ROM-blok 11-2
 ROM-chip 7-1
 RUN 2-6, 3-4
 Radialen 6-1
 Randaarde 2-1
 Random access 9-14 ev.
 Random getal 6-2
 Record 9-14
 Register 8-10, 8-13
 Reistijden 3-5
 S
 SVX-4 9-16
 Schakelen 2-7
 Schermbeschrijving 8-10
 Sectoren 9-2
 Sequentieel 9-12
 Shape-programma 7-17
 Shapetable 7-10 ev.
 Slot 5-25, 10-1 ev., 10-10
 Software 2-16, 13-4
 Spatiebalk 2-9
 Spaties 3-9
 Stack 5-29
 Statement 2-16, 5-19
 Storing 12-1, 12-3
 String concatenatie 5-15
 String variabelen 3-8
 Strings 3-9, 5-11
 Structuur 2-19
 Subroutines 8-1
 Subscript 5-13
 Syntax 2-10, 2-18
 Systeem commando's 5-1 ev.
 T
 Tab-fields 2-5, 3-5
 Tekstcontrole 6-6
 Tekstcorrectie 2-8, 2-10
 Tekstvenster 3-4, 5-10, 7-6, 8-8
 Temperatuur 10-4
 Text-display 8-8 ev.
 Timing 11-1 ev.
 Toets 12-3
 Tracks 9-2
 U
 V
 Variabelen 3-6
 Verschillen 4-4
 Video output 5-12, 10-2
 Volumeknop 10-5
 Volumenummer 9-17
 Voorrangsregels 3-11
 W
 Wedstrijd-score 5-19
 Woorden, gereserv. 4-2, 6-11
 X
 Y
 Z
 Zoekprogramma 5-3

HOOFDSTUK 2 HOE TE BEGINNEN

1. Het systeem opstarten

Het APPLE-II Personal Computersysteem (PC)

Bekijk het.



Oppervlakkig ziet het eruit als een moderne typemachine. Een blik onder de bovenklep is echter voldoende , om te ervaren , dat hier een krachtige en geavanceerde computer schuilgaat. Til voorzichtig de achterzijde van de bovenklep op. Met een zachte klik komt hij los. Schuif de klep wat naar achter en neem hem weg.

Aan de binnenkant van het systeem bevindt zich een ordelijk geheel van zeer moderne elektronische componenten.

Daarin mag niets rammelen , of loszitten.

Breng de bovenklep weer op zijn plaats en klik hem vast.

Controleer nu de achterzijde van de APPLE-II .

De aan/uit schakelaar moet in de "off"-positie staan , terwijl de netspanningsschakelaar het juiste voltage (bijv. "220") moet aangeven. Evenals andere elektronische apparaten sluit je de APPLE-II met een van randaarde voorziene stekker op het lichtnet aan. De randaarde is een veiligheidsvoorziening.

Verbindt de APPLE-II vervolgens met het televisietoestel of data-monitor.

Het is eventueel mogelijk kleuren toe te passen , maar dan moet je een speciale "PAL interface kaart", "RGB kaart", of NTSC-monitor gebruiken.

Door de aan/uit schakelaar aan de achterzijde van de APPLE-II te gebruiken , wordt het systeem ingeschakeld. Het verklikkerlampje onder de toets , genaamd "power" , gaat nu branden.

Die toets heeft verder geen schakelfunctie.

Het volgende moet nu gebeuren: de luidspreker van de APPLE-II zal een welluidende "biep" laten horen en links onder in het televisie-beeld verschijnt hierna een merkwaardig gevormd haakje (7) , waarnaast een aan en uit knipperend vierkantje zichtbaar is.

Bij het APPLE-II type zonder zgn. autostart ROM dient eerst de CTRL-toets tezamen met de B-toets te worden ingedrukt , gevolgd door de RETURN-toets.

Dit blinkende vierkantje noemt men de "loper" , of in computer-taal: de cursor. Wanneer je een paar keer op de brede spatiebalk , onder in het toetsenbord , drukt , zie je de cursor opschuiven - hij "loopt" over het beeld.

Afgezien van het haakje en de aanvallig knipperende cursor kunnen er bij het inschakelen van de APPLE-II zonder autostart ROM allerlei vreemde tekens op de beeldbuis zichtbaar zijn. Dit rommeltje kunnen we op een eenvoudige wijze kwijtraken!

Je drukt even op de toets , genaamd ESC (van ESCape=ontsnappen aan) , en daarna op de SHIFT-toets , die tegelijk met de P-toets wordt gebruikt. Nadat SHIFT-P werd ingedrukt drukken we op de RETURN-toets - en weg zijn alle vreemde inschakeltekenen. Haakje en cursor staan keurig boven in het beeld.



Het haakje is een van de zogenaamde "prompt" tekens , die de APPLE-II kan tonen. De prompt verradt de taal , waarin de computer aangesproken wenst te worden. Hij is dus min of meer onze souffleur , die je influistert in welke taal je moet spreken.

Als de prompt en de cursor (soms ook de cursor alleen) in het beeld zichtbaar is , bedoelt de APPLE-II dat het jouw beurt is en dat hij via het toetsenbord graag met je in contact treedt. M. a. w. het is nu jouw beurt! Ga je gang.

LET OP!!!

Valt de prompt en de cursor gedeeltelijk , of zelfs geheel buiten beeld , dan zal het beeldvenster van de televisie bijgesteld moeten worden.

De prompt, die nu zichtbaar is (□), geeft aan dat door de computer APPLESOFT Floating Point Basic wordt gesproken. En gemakshalve houden we ons daar maar aan...

2. Iets over Commando's

Voordat de APPLE-II iets voor je kan doen , moet je hem vertellen , wat er precies wordt verlangd.
De APPLE-II vraagt om ondubbelzinnige instructies , die in een juiste volgorde moeten worden gegeven.
Met onze spreektaal lukt dat gewoon niet.
Neem bijvoorbeeld het bevel "Geef acht!"-moet de computer nu opletten voor wat hierna komt , of moet hij het getal 8 afbeelden? Aan spreektaal heb je dus niets.
Vandaar dat je een speciale programmeertaal moet gebruiken.
APPLESOFT is zo'n hogere programmeertaal.

Type nu eens het volgende:

2+3

en druk vervolgens op de RETURN-toets. In een oogwenk zijn prompt en cursor weer in beeld , echter een regel lager. Verder gebeurde er niets. Verwachtte je het antwoord van deze rekensom?

De APPLE-II kreeg ondubbelzinnige instructies , om in zijn geheugen 2 met 3 te vermeerderen. Hij deed dit en gaf in een oogwenk de beschikking over het toetsenbord terug.
Dat was niet helemaal de bedoeling. We wilden immers het resultaat van de rekensom zien?

Type nu:

PRINT 2+3

en druk op de return-toets.

Dit keer ontving de APPLE-II ondubbelzinnig opdracht , om 2 met 3 te vermeerderen en het resultaat te laten zien (PRINT=druk af).

Type nu:

PRINT 1 , 2;3 , 4

en druk om een APPLE-II antwoord te krijgen op de return-toets. Dat is niet mis!

1 23 4

Het is je duidelijk , dat het gebruik van de komma(,)en punt-komma(;) te maken heeft met de verdeling van de getallen over het beeld.

Dat is een juiste conclusie , die overigens van veel betekenis is. De komma(,) wordt namelijk gebruikt , om het beeld in kolommen te verdelen. De 40 leestekens , die op een regel kunnen , worden erdoor in 3 kolommen verdeeld. Een kolom heet een "tab field". Onder leestekens verstaan wij letters , tekens , getallen en spaties. Misschien is het wel beter , om niet meer te spreken over leesteken , maar over karakter.

LET OP !!!

Waar je gewoon bent , om getallen "achter de komma" te schrijven , gaat dat niet bij de Personal Computer. Daarvoor wordt de punt (.) gebruikt. De APPLE-II kent geen "drijvende komma" , maar een vaste punt , die gebroken getallen verdeelt!

Type vervolgens:

```
PRINT 2+3 , 1.5*2.5
```

Als de opdracht klaar is , gebruik je steeds de return-toets , waardoor de APPLE-II weet dat het nu zijn beurt is.

Wat zien we nu op het beeldscherm verschijnen?

```
5          3.75
```

We gaven de APPLE-II instructies , om 2 met 3 te vermeerderen , vervolgens een "tab field" op te schuiven , 1. 5 met 2. 5 te vermenigvuldigen en het eindresultaat af te beelden.

Om te vermenigvuldigen werd het (*) teken ingevoerd , om misverstanden met de letter x te vermijden. Hetzelfde is het geval met cijfer 0 en de letter o. Het zijn eigenwijze dingen , die computers. . . .

Het gaat erom , op ondubbelzinnige wijze te laten weten , wat er nu van de APPLE-II wordt verlangd.

Tot zover heb je de computer commando's gegeven , die tot een onmiddellijke reactie leiden. Je had de return-toets nog maar amper losgelaten , of de APPLE-II was al klaar met zijn opdracht.

Een veel interessanter soort instructies behoort tot de groep van de INDIREKTE COMMANDO'S. De indirecte commando's worden voorafgegaan door een regelnummer. Een logisch geheel van indirecte commando's heet een programma.

Type dit eens:

```
10 PRINT "APPELTJE"(return)
20 PRINT APPELTJE,2 + 3.7
30 END
```

Iedere keer verschijnt de prompt en de cursor een regel lager .

Door de return-toets geven we aan , dat de betreffende programmaregel klaar is .

Wanneer het programma gereed is , nemen we een commando op, die de APPLE-II vertelt , dat we helemaal klaar zijn - end (= einde) .Om het programma nu te laten werken , type je zonder regelnummer

RUN

Oh , oh , wat krijgen we nu ?

```
APPELTJE
0      5.7
```

Merkwaardig ? Natuurlijk niet ! De " " in regel 10 zorgen ervoor , dat wat tussen die aanhalingstekens staat , later letterlijk wordt weergegeven .

In regel 20 echter worden geen aanhalingstekens gebruikt en begrijpt de APPLE-II , dat het woord APPELTJE bedoeld is , om een rekenkundige variabele mee aan te duiden .

Zo van APPELTJE = 4 (A + B = 6 , weet je nog ?) . Omdat APPELTJE geen waarde meekreeg , gaf de APPLE-II aan het woord de waarde 0 .

Om het oorspronkelijke programma weer op het beeldscherm te krijgen , type je - zonder regelnummer - het volgende:

LIST (return)

en jawel !

```
10 PRINT "APPELTJE"
20 PRINT APPELTJE,2 + 3.7
30 END
```

Je vraagt dus gewoon even om de lijst (LIST) . . . Wil je het programma nog eens bekijken , geef je een RUN. RUN betekent zoveel als " laat lopen " !

Wil je een nieuw programma maken , dan kan het nodig zijn, om een oud programma uit te wissen en uit het geheugen te verwijderen. De APPLE-II even uit en aan zetten kan hiervoor een oplossing zijn .Maar het is geen beste. Het kan bepaald minder hardhandig. Je kunt de oude regelnummers overtypen (zelfs al vul je er niets achter in). De oude zijn dan onherroepelijk verdwenen .

De beste werkwijze is het typen van het woord

NEW

De APPLE-II kent dit woord en wist het geheugen in afwachting van een geheel nieuw programma .
Type daarom :

NEW (return)
LIST (return)

Kijk , kijk , er komt geen listing (programma lijst) meer , want het oude programma is na het statement NEW onherroepelijk vervallen .

Het is wellicht goed , om er hier even op te wijzen , dat het veelvuldig AAN en UIT schakelen van de APPLE-II Personal Computer in feite overbodig is .

Hoewel de APPLE-II geen nadelige gevolgen ondervindt van het aan en uit schakelen , lijkt het aan te bevelen , om het systeem telkens zo lang mogelijk onder stroom te houden .

Bij het uitschakelen verlies je programma ' s en taal . Alleen de startprocedure brengt je weer in de juiste taal terug .

De APPLE-II Personal Computer is uiterst energie - vriendelijk .

Het energieverbruik ligt op het niveau van een moderne , doorsnee transistorradio en het is uit dat oogpunt niet zinvol en noodzakelijk door veelvuldig te schakelen energie te sparen .

3. Programma ' s corrigeren

Op het toetsenbord van de APPLE-II komen toetsen voor , die een bijzondere functie hebben. Enkele ervan hebben een speciale besturingsfunctie , andere dienen om de invoer van karakters (letters , cijfers , tekens en spaties) te vergemakkelijken. Laten wij nu enkele bijzondere toetsen nader bestuderen .

a. ESC toets

Links in het toetsenbord vinden we de ESC toets.
ESC is de afkorting van Escape , d.w.z. " ontsnappen aan,"

Al eerder kwamen we de toets terloops tegen , toen hij in combinatie met de P - toets werd gebruikt (ESC , Shift - P , return) , om het beeldscherm schoon te vegen . De ESC toets wordt weliswaar in combinatie met een andere toets gebruikt , maar hij wordt ermee nooit tegelijk ingedrukt .

Eerst de ESC - toets , daarna de andere .

Voor tekstcorrecties zijn 4 ESC combinaties beschikbaar :

ESC A beweegt de cursor 1 plaats naar rechts
ESC B beweegt de cursor 1 plaats naar links
ESC C beweegt de cursor 1 plaats naar beneden
ESC D beweegt de cursor 1 plaats naar boven

Door de ESC combinaties verschillende malen achtereen te gebruiken , kan elke plaats op het beeldscherm met de cursor worden bereikt , zonder dat het geheugen erdoor wordt beïnvloed .

De computers die aangeduid zijn met APPLE-II PLUS of APPLE-II EUROPLUS , beschikken naast de bovengenoemde besturingsfuncties nog over extra zgn. " edit " functies m. b. v. de ESC I , J , K en M toetsen .

ESC K beweegt de cursor 1 plaats naar rechts
ESC J beweegt de cursor 1 plaats naar links
ESC M beweegt de cursor 1 plaats naar beneden
ESC I beweegt de cursor 1 plaats naar boven

Het grote verschil is dat je bij deze cursor - controle slechts 1 keer de ESC - toets hoeft in te drukken en in willekeurige volgorde de K , J , M en I kan bedienen totdat je op de gewenste positie op het beeldscherm bent.

Druk dan op een willekeurige toets (behalve I , J , K , M , CTRL , ESC en RETURN) om uit deze " edit - mode " te komen .

b. ← toets

Deze toets bevindt zich rechts van het toetsenbord .
Hij beweegt de cursor telkens 1 plaats over het
beeldscherm naar links .
Er is echter een interessant verschil met de ESC
combinaties .
Wanneer je de ← toets naar links over een tekstregel
beweegt , wordt er telkens 1 karakter van die regel uit
het APPLE-II geheugen gewist ! Maar het is niet mogelijk
, om de meest linkse positie van de regel te bereiken -
daarvoor moet je de ESC B combinatie gebruiken .

c. → toets

De toets verplaatst de cursor (goed geraden !) telkens
1 positie naar rechts. Ook hier een interessant verschil
met de ESC combinaties. Bij elke verplaatsing naar rechts
door de cursor , mits uitgevoerd d. m. v. de → toets ,
wordt 1 karakter uit de regel in het geheugen gekopieerd.

De toepassing hiervan bespreken we direkt .

d. REPT toets

Deze toets vinden we naast de RETURN - toets. De
afkorting REPT komt van REPEAT , hetgeen " herhaal "
betekent .
Het is daarom duidelijk duidelijk , wat voor gevolgen het
onverhoeds indrukken van deze interessante toets heeft !

De REPT toets wordt tegelijk met een andere toets
ingedrukt .
Het gevolg is , dat het gekozen karakter zolang wordt
herhaald , tot die toets , de REPT toets , of beide
worden losgelaten .

Type :

X

en houdt de X toets ingedrukt. Druk er nu de REPT toets
bij in en rrrroeoetsssch ;

XXXXXXXXXXXXXXXXXXXXXXXXXXXXX....

totdat je een of beide toetsen loslaat. Bedenk eens , wat
er precies gebeurt , wanneer je de lange spatiebalk
(onder in het toetsenbord) tegelijk met de REPT toets
ingedrukt houdt .

Is de APPLE-II niet veelzijdig ?

e. corrigeren van karakters in een regel

Je kunt er niet oohe..... , oeps , omheen , dat er typefouten worden gemaakt. Het zal daarom nogal eens voorkomen , dat je te maken krijgt met tekstcorrectie. In het computertaaltje noemt men dit " editing " . In een aantal gevallen zal de APPLE-II Personal computer hiervan tijdens de uitvoering van het programma mededeling doen. Er volgt een spontane foutmelding (zie aldaar) en het programma wordt onderbroken . Zo'n foutmelding wordt voorafgegaan door een waarschuwende " biep " - toon uit de luidspreker . Laten we eens een fout intypen :

```
10 PLINT "APPLE-II"
```

vervolgens typen we RUN en " biep " :

```
?SYNTAX ERROR IN 10
```

We zien dus dat er in regel 10 een fout werd gemaakt en dat die fout de woordbouw (van de instructie) betrof . Het woord SYNTAX heeft bij computers meer betrekking op de opbouw van het statement (het commando , vaak ook de instructie) , dan op de totale zinsbouw .

LET OP !

Vanaf hier zullen we het gebruik van de return - toets niet verder specificeren. Zodra een opdracht , of programmaregel klaar is , drukken we op de return - toets.

Terug naar onze opzettelijke fout . Door LISI 10 te typen , halen we de foute programmaregel weer uit het geheugen op :

```
LIST 10  
10 PL INT "APPLE-II"
```

Om de fout nu te herstellen , wat noodzakelijk is voor een verdere programma uitvoering , kunnen we 2 dingen doen :

- a. We typen de gehele regel 10 over. Dat is omslachtig , maar natuurlijk wel afdoend .
- b. We corrigeren de fout. Dat gaat heel goed met de hiervoor reeds besproken , bijzondere toetsen .

Hoe pakken we dit aan ?

Om te beginnen brengen we de cursor over het eerste karakter van de regel 10. Dat is dus het cijfer 1 . Je drukt net zo vaak op ESC D tot de cursor op de regel is aangekomen en gebruiken eventueel ESC B , om de " loper " over de 1 te krijgen. M. a. w. staan 1 en cursor nu in hetzelfde " vakje " .

Computergebruikers met de Autostart-ROM kunnen dus 1 keer ESC indrukken en dan met de I en de J de cursor positioneren. De cursor geven we hier aan met een * .

```
*0 PL INT "APPLE-II"
```

Nu drukken we een paar keer op de → toets , totdat de cursor in hetzelfde vakje als de fout staat :

```
10 P* INT "APPLE-II"
```

Op deze plaats typen we nu het juiste karakter - r .

```
10 PR*INT "APPLE-II"
```

Dan bewegen we de cursor d. m. v. de → toets zover naar rechts , dat alle karakters van de regel opnieuw in het geheugen worden gekopieerd. Tenslotte drukken we de return - toets in (we zijn immers klaar) .

Regel 10 is nu hersteld en een RUN van dit programma bewijst het succes van ons werk .

Type vervolgens nogeens :

```
LIST 10  
10 PRINT "APPLE-II"
```

De reden , waarom het foutieve woord PLINT door de computer werd " opengebrokeu " , is de herkenning van het gereserveerde woord INT (zie aldaar) . Hierop gaan we nu niet verder in .

Even gemakkelijk gaat het tussenvoegen van tekst . Je hoeft dus niet in paniek te raken , wanneer mocht blijken , dat er een stuk in een programmaregel is vergeten .

Stel dat we willen typen " HET APPLE-II PERSONAL COMPUTER SYSTEEM " . Terwijl we hier mee bezig zijn komt moeder de vrouw met de koffie binnen. Hierdoor afgeleid weten we niet meer precies waar we waren en met het commando LIST vragen we de ingevoerde programmaregel op :

```
LIST 10  
10 PRINT " HET APPLE-II SYSTEEM "
```

Tja , dat was niet geheel de bedoeling. We moeten dus nog wat tussenvoegen .

Om deze fout te herstellen , brengen we de cursor opnieuw over het eerste karakter van de programmaregel .

```
*0 PRINT " HET APPLE-II SYSTEEM "
```


Vervolgens laten we de cursor zover naar rechts lopen , tot de cursor op het punt is aangekomen , waar we het mankerende tekstgedeelte willen tussenvoegen .

```
10 PRINT " HET APPLE-II * SYSTEEM "
```

Door de cursor naar rechts te laten bewegen , hebben we de gepasseerde karakters in het geheugen gekopieerd . Daar zijn we dus vanaf. Maar nu ?

Nu moeten we even goed nadenken en opletten ! Door een ESC D brengen we de cursor 1 plaats omhoog en krijgen nu ruimte , om het missende tekstgedeelte uit te typen. Maar let op : eerst 1 spatie invoegen - anders zal de tekst straks aan elkaar kleven ! We zien nu :

*

```
10 PRINT " HET APPLE-II SYSTEEM "
```

Type nu de mankerende woorden PERSONAL COMPUTER .

PERSONAL COMPUTER

```
10 PRINT " HET APPLE-II SYSTEEM "
```

Haal nu de cursor met ESC B weer terug tot het punt , waar we aanvankelijk waren begonnen :

* PERSONAL COMPUTER

```
10 PRINT " HET APPLE-II SYSTEEM "
```

De cursor moet nu weer naar beneden (ESC C) en met de → toets kopiëren wij de rest van de programmaregel in het geheugen. En dan (voor het laatst) return .

Na een LIST 10 moet de gecorrigeerde regel er zo uitzien:

```
10 PRINT " HET APPLE-II PERSONAL COMPUTER SYSTEEM "
```

Vraag : wat zou er gebeurd zijn , wanneer we na het uittypen van de missende woorden de cursor met de < toets naar links terug hadden bewogen tot boven de plaats, waar we waren begonnen ?

Als je dit begrepen hebt , is de " editing " geen enkel probleem meer !

f. het corrigeren van een ongelukje

Misschien is het al gebeurd ! Per ongeluk de RESET toets ingedrukt , in plaats van de RETURN .
Het gevolg was nogal spectaculair , nietwaar ?
De prompt (wat was dat ook al weer voor ding ?) veranderde in een * .
M. a. w. schakelde de APPLE-II over op een andere taal , in dit geval de " monitor " (zie aldaar) .
Om hem weer in APPLESOFT terug te krijgen , moest je toen de computer uit en aan zetten .
Niet aardig , om de dienstbare APPLE-II zo hardhandig aan te pakken voor een zelfgemaakte fout !
Dat kan ook anders .
Mocht zich deze "vergrijping" nogeens herhalen , druk je slechts CTRL-C , return , om de juiste taal weer terug te krijgen en vernietiging (wissen) van het lopende programma te voorkomen .
Mocht deze procedure , om wat voor reden dan ook , niet gelukken , druk je CTRL-B , return , om weer in APPLESOFT te komen , maar..... helaas , helaas het lopende programma is dan vernietigd ("gekilled" zeggen deskundigen) .

Het indrukken van de RESET - toets heeft geen gevolgen voor APPLE-II PLUS en EUROPLUS computers. De cursor keert terug in APPLESOFT (of Integer - Basic , afhankelijk van de gebruikte taal) en het programma kan ge-LIST en ge-RUNd worden. De laatst ingevoerde regel is echter niet aan het programma toegevoegd , waardoor het RUNnen van het programma vreemde verschijnselen te zien kan geven.

Recentere versies van de APPLE-II zijn uitgerust met een "Encoder Board". Dit board stelt je in staat op twee manieren van de RESET - toets gebruik te maken.
De eerste manier is de "standaard" functie van RESET. Voor diegenen die wat angstig zijn voor het indrukken van deze toets , of liever niet direkt met een leeg scherm geconfronteerd willen worden na het (per ongeluk) indrukken van de toets , is er een tweede functie.
Deze kan geselecteerd worden door de schakelaar op het "Encoder Board" zo te zetten , dat deze gelijk valt met het witte rechthoekje.
Na deze modificatie zal je APPLE-II alleen reageren op een CTRL - RESET , die tegenwoordig kan worden door zowel de CTRL - als de RESET - toets in te drukken.
Kijk je documentatie na of jouw APPLE van dit board voorzien is !

g. weghalen van een ingevoerde regel

Het komt vaak voor , dat je een regel aan het invoeren bent , die bij nadere bestudering niet correct blijkt . Stop met verder typen en druk CTRL - X . De regel wordt dan direkt gewist en je kunt overnieuw beginnen . Probeer het volgende , wat verder gaande programma :

```
10 PRINT "IK BEN"  
20 PRINT "APPLE-II"  
30 PRINT "PERSONAL COMPUTER"  
40 PRINT "DE BAAS"
```

Wanneer je van mening bent , dat we teveel opscheppen over de APPLE-II Personal Computer , haal je gewoon regel 20 en 30 weg. Dit kan op twee manieren :

1. Type :

```
20  
30
```

Daarmee zijn die regels " leeg gemaakt " , spelen ze verder in het programma geen rol en worden ze door de APPLE-II verwijderd .

2. De beste manier is gebruik te maken van het statement DEL .

Dit , wat aanstootgevend statement is de afkorting van DElete (= laat weg) .

Type daarom :

```
DEL 20,30  
RUN  
IK BEN  
DE BAAS
```

Oh zo ! Probeer nu regel 10 met het DEL statement uit het programma te verwijderen. Mocht dat niet lukken , probeer het dan nog eens op de eerst genoemde manier. Succes !

Overzicht van alle speciale control en editing karakters

RESET Onderbreekt direct elk lopend programma en RESET de computer. Het systeem laat een toontje horen, komt in de Monitor , en geeft * als prompt .
(NIET BIJ COMPUTERS MET AUTO START-ROM)
Het gebruik van de RESET toets vernietigt geen bestaande BASIC of machinetaal programma ' s .

CTRL-B Vanuit de Monitor (promptkarakter *) zal , CTRL-B, return overgang naar APPLESOFT tot gevolg hebben.
Elk bestaand BASIC programma wordt " gekilled " ; HIMEM wordt op de hoogste RAM lokatie gezet
(zie aldaar) .

CTRL-C Vanuit BASIC stopt dit commando de programmaloop, en meldt : BREAK IN XXXX .
Vanuit de monitor zal CTRL-C , return overgang naar APPLESOFT tot gevolg hebben , zonder dat bestaande programma ' s worden vernietigd .

CTRL-G Activeert de luidspreker .

CTRL-H Plaatst de cursor 1 plaats naar links, verwijderd overschreven karakters uit het geheugen , maar niet van het scherm. Gelijk aan ← toets .

CTRL-J Nieuwe regel (Line Feed) .

CTRL-V Complement van CTRL-H. Plaatst cursor een positie naar rechts en kopieert karakter van scherm naar geheugen . Gelijk aan → toets .

CTRL-S Laat een LISTING stoppen en hervatten
(Alleen bij APPLE-II PLUS en EUROPLUS)

CTRL-X Verwijdert lopende regel van scherm en uit systeem .

ESC A , K Cursor 1 plaats naar rechts

ESC B , J Cursor 1 plaats naar links

ESC C , M Cursor 1 plaats naar beneden

ESC D , I Cursor 1 plaats naar beneden

ESC E Wist het scherm vanaf de cursor tot het einde de regel

ESC F Wist het gehele scherm vanaf de cursor

ESC , SHIFT-P Wist het hele scherm en plaatst de cursor boven in het scherm

LET OP :

SHIFT-X Druk SHIFT toets in , druk X in , laat X los , laat SHIFT los .

CTRL-X Druk CTRL toets in , druk X in , laat X los laat CTRL los .

ESC X Druk ESCAPE toets in , laat ESCAPE los , druk X in ,laat X los .

REPT X Druk X in , druk REPT in , laat REPT los , laat X los .

4. Wat is APPLESOFT ?

Een APPLE zonder programma is als een auto zonder benzine.

Alle onderdelen zijn wel aanwezig , maar er is nog 'iets' nodig , om het geheel te laten werken. Voor de auto is dat 'iets' benzine. Voor de APPLE-II is het de programmatuur , die men vaak " software " noemt .

Als je je nieuwe wagen uit de garage ophaalt , zorgt de garagehouder er wel voor , dat er wat benzine in de tank zit .

Met de APPLE-II Personal Computer is dat niet anders ; je krijgt hem met een hoeveelheid ingebouwde software
In tegenstelling tot de benzine , raakt de computer-software niet op en is hij telkens opnieuw beschikbaar !

De ontwerper van de APPLE heeft de software ondergebracht in heel kleine elektronische componenten , die men PROM's noemt .

De PROM is een (voor de ontwerper) programmeerbaar uitleesgeheugen. Hij heeft er dus wat software ingestopt , die wij er - om het zo te zeggen - slechts kunnen uithalen. Deze software maakt het je gemakkelijk , om bepaalde programma ' s te ontwikkelen .

Het schrijven van programma's voor de computer kan op verschillende manieren gebeuren. We kunnen het in machine - code doen , in het tweetallig stelsel. Deze binaire werkvorm is moeilijk en geeft een grote kans op later nauwelijks terugvindbare fouten .

Overigens zijn wij wel gewend aan het tientallig stelsel, dat loopt van 0 tot 9. Je weet toch nog , dat het getal 10 de combinatie is van 1 en 0 ? Nee , nee , zo lang is dat nog niet geleden

Een stap verder is het schrijven in assembly. De assembly - language (language = taal) is een symbolische weergave van machine instructies. De kans op het maken van fouten is misschien kleiner , het werken met de symbolische machine - instructies (mnemonics) is verre van gemakkelijk .

Omdat de instructies ook in dit geval tweetallig gecodeerd in het geheugen moeten komen , is er een vertaal programma nodig. Dit programma noemen we een assembler. Nog een stap hoger vinden we computertalen , die de spreektaal steeds dichter gaan benaderen. In deze hogere programmeertalen is een enkel ' statement ' (= direkt uit te voeren commando , of een bewerkingsinstructie) vaak gelijk aan meerdere machine - instructies .Het zal daarom duidelijk zijn , dat het vertaalprogramma hierbij veel ingewikkelder is , dan de assembler .

Er zijn 2 typen vertaalprogramma ' s voor hogere programmeertalen : de compiler en de interpreter . Een compiler (spreek uit : kompijler) vertaalt het programma uit een hogere programmeertaal in zijn geheel naar machine - code en voert dit vervolgens uit . Een interpreter (spreek uit : inturprittur) vertaalt steeds een stukje uit het programma en voert dat eerst uit , om weer verder te gaan. Bij programma ' s , die door een interpreter vertaald moeten worden , staan de gegevens in een niet - vertaalde vorm in het geheugen. Het grote voordeel hiervan is , dat je een haast intelligent kontakt met de computer kunt hebben. Als een statement wordt ingevoerd , vertaalt de interpreter dit en kan hij het direkt uitvoeren , of een onmiddellijke foutmelding laten volgen. Je kunt op deze wijze de programmeertaal snel onder de knie krijgen . Een nadeel is , dat een programma - onderdeel , dat verschillende malen moet worden uitgevoerd , ook verschillende malen moet worden vertaald . De APPLE-II is uitgevoerd met een uitgebreide BASIC Interpreter . Voorts is hij voorzien van een assembler , een disassembler voor terugvertaling , en een systeemmonitor voor het ontwikkelen en testen van programma ' s in machinetaal . Om je te laten weten , welk gedeelte van de systeemsoftware op het moment actief is , werd de prompt ingevoerd . De prompt geeft aan , welke taal er wordt gesproken en bij het invoeren van gegevens wordt verwacht. Als je aan de beurt bent , om iets in te voeren , staat de prompt links aan het begin van de regel.

Voor monitor is dat een	* .
Voor APPLESOFT	[.
Voor INTEGER BASIC	> .
Voor assembler	! .

Door CALL -151 kom je van APPLESOFT op een kalme manier in monitor .

Door F666G te typen kom je van monitor in assembler , indien er een zgn. Integerkaart in de APPLE-II is gemonteerd .

Tenslotte kom je met RESET , C081 uit monitor in Integer Basic (niet standaard aanwezig bij de APPLE-II PLUS). Gewoon mee experimenteren , maar zorg dat je hierna weer terug bent in APPLESOFT , omdat we het daarover verder zullen hebben .

APPLESOFT is een zeer uitgebreide BASIC programmeertaal .
BASIC kwam kort na 1960 in de Verenigde Staten tot
ontwikkeling en betekent " voor Beginners geschikte
Algemene Symbolische Instructie Code ". Behalve de BASIC
commando ' s en instructies , die algemeen toegepast
kunnen worden , kent APPLESOFT een aantal
machine-gerichte eigenschappen .
Dat zijn de mogelijkheden voor het werken met grafieken
in een grof en zeer fijn raster , de 4 analoog / digitaal
kanalen bijvoorbeeld voor spelletjes en geluid .
Nu is het overigens niet zo moeilijk , om geluid te maken
met de APPLE-II .

Je kunt er met je vingers op trommelen , met je nagels
aan krabben en je kunt hem natuurlijk uit je handen laten
vallen.... Wij bedoelen echter het opwekken van geluid
met APPLESOFT instructies .

In het hierna volgende hoofdstuk gaan we in op het
gebruik van de BASIC programmeertaal , zoals die met
voorbehoud van enkele verschillen , op de meeste
computers met een BASIC Interpreter voorkomt .

Natuurlijk komen ook de specifieke APPLE-II uitbreidingen
aan de orde. Het gaat daarbij niet om een
programmeercursus , maar een overzicht van de vele
beschikbare statements met hun syntactische opbouw. De
statements behoren nu eenmaal op een bepaalde wijze te
worden geformuleerd. De syntax moet voor de APPLE-II
foutloos en begrijpbaar zijn , wil een commando , of een
verwerkingsinstructie uit een programma op een
tevredenstellende manier worden uitgevoerd .

Genoeg met al dit moeilijks .

Het gaat erom , dat je prettig en efficiënt met je
Personal Computer werkt. Daarvoor bestaan er dus
'spelregels ' en daarin gaan we ons straks verdiepen.
Probeer zoveel mogelijk uit en wees niet bang dat er iets
kapot zal gaan .
Tenminste zolang je slechts probeersels via het
toetsenbord binnen de computer brengt .

Aan de prompt kun je zien , of je in APPLESOFT bent. Het
' in ' een bepaalde taal ' zijn ' , behoort tot het
computer - jargon , we beginnen nu langzaam aan ingewijd
te raken .

Voor we gaan beginnen met het volgende hoofdstuk nog een paar raadgevingen. Het kan gebeuren , dat je iets ingevoerd hebt , waardoor de APPLE-II geen zinvolle programma ' s meer uitvoert , maar weigert het toetsenbord aan je terug te geven. We noemen dat :
" Het systeem hangt !"
Zo ' n toestand hef je altijd op met een RESET !
Om het programma weer terug te krijgen , type je :

CTRL-C , return

Ook voor het onderbreken van een lopend programma gebruik je :

CTRL-C , return

Zit de zaak op alle mogelijke manieren vast en ' hangt ' het systeem , dan rest slechts :

RESET , CTRL-B , return

Nogmaals , dit is de opstart-procedure en je wist bestaande programma ' s , maar komt in APPLESOFT terug .

Bij de beschrijving van de structuur van APPLESOFT commando ' s worden bepaalde tekens gebruikt. Deze geven symbolisch nader uitleg over de syntactische definiering. M. a. w. je commando ' s MOETEN juist samengesteld , of gedefinieerd , zijn .

Accolade { en } wordt gebruikt , om aan te geven , dat wat er tussen staat herhaald mag worden. Rechte haakjes [en] geven aan , dat wat er tussen staat mag , maar niet moet , voorkomen. Onze gewone haakjes (en) geven aan , dat ertussen parameters MOETEN worden opgegeven .

expr	in commando's betekent , dat elke willekeurige expressie mag worden ingevuld - dus rekenkundige, string , en integer expressie (dat laatste wijst zich vanzelf) .
aexpr	betekent rekenkundige expressie
sexpr	betekent string expressie
rnum	betekent regelnummer
avar	betekent array variabele
svar	betekent string variabele
var	betekent rekenkundige , string of integer variabele
string	betekent string
lit	betekent literal
real var	is een real variabele
enz .	

Neem hier gewoon kennis van ; later wordt de betekenis vanzelf aanmerkelijk duidelijker .

Als van een expressie of variabele de absolute waarde
wordt bedoeld , dan wordt dit aangegeven door / / .
De waarde van variabele 1 wordt dan /var 1/ .

HOOFDSTUK 3 PROGRAMMEREN IN BASIC

1. Het getallensysteem

Om te begrijpen , wat de Personal Computer met de hem aangeboden getallen doet , is het handig even wat vroegere schoolherinneringen bij je op te halen. Het zal je toen niet zijn ontgaan , dat er tijdens de rekenles werd gesproken over ' gehele getallen ' en over ' gebroken getallen '.

Tot de gehele getallen behoren 2 , -3 , 255 , -32767.

Gebroken getallen zijn 1.75 , -0.162.

Om dit niet onnodig ingewikkeld te maken , schuiven we de soms moeilijk te bevatten ' algebraïsche irrationele ' zoals $\sqrt{2} = 1.414$ en de ' transcendent irrationele getallen ' , zoals $\ln 2 = 0.693$ en $e = 2.718$ ook onder de noemer van de gebroken getallen. We vermelden dit maar even , om boze reacties te vermijden. Gehele getallen en gebroken getallen.

Dat vonden de ontwerpers van BASIC ook , toen zij hun ' real precision ' en ' integer ' getallen invoerden. Het gaat daarbij om de nauwkeurigheid van de getallen. Het merkwaardige doet zich hierbij voor , dat de laatste jaren in steeds sterker wordende mate ten onrechte het begrip ' integer ' in onze taal is binnengeslopen , als aanduiding voor het gehele getal. Het is ook zo'n mooi , kort woord ; maar het hoort eigenlijk niet in onze taal thuis.

De integer is een geheel getal.

Het ' real precision ' getal is het gebroken getal.

Zoals je eerder hebt gezien , wordt er geen getal achter ' de komma ' genoteerd , omdat de komma (,) een rol speelt bij de regelverdeling , ook wel het print-formaat genoemd.

Om de gebroken getallen te noteren , gebruiken we de punt (.).

Dit verklaart het enthousiasme voor het fraaie begrip 'floating point ' , dat vaak in computerboeken opduikt.

De APPLE-II kan inwendig werken met een grotere nauwkeurigheid , dan 8 getallen achter de..... punt. Dat heeft een aantal interessante gevolgen. Je zult even moeten wennen aan de wijze , waarop Personal Computers de getalsprecisie op het beeldscherm brengen.

Gehele getallen moeten liggen tussen -32767 en 32767.

Gebroken getallen moeten liggen tussen -10^{38} en 10^{38} .

Het ^ teken betekent macht.

De ' floating point ' notering ziet er nu als volgt uit :

+ 1.11111111 E + 00

Op het eerste gezicht wat raadselachtig , maar dat valt mee.

Eerst komt het teken (+ of -) van het getal , dat overigens bij positieve (+) getallen niet wordt weergegeven. Voor de punt wordt 1 getal afgedrukt , dat niet 0 moet zijn.

Anders wordt daar geen getal voor de punt geplaatst. Vervolgens komt een mantisse van 8 cijfers na de punt. De E duidt erop , dat het afgebeelde getal tot een macht van 10 verheven moet worden , waarvan het teken (+ of -) gevolgd moet worden door die exponent.

Bijvoorbeeld :

$1 \cdot 10^{20}$ wordt : 1E+20

De APPLE-II laat overbodige nullen (0) in de mantisse wegvallen , en als er na de onderdrukking van nullen helemaal geen mantisse meer overblijft , wordt natuurlijk geen decimale punt afgedrukt.

Je mag maximaal 38 cijfers via het toetsenbord invoeren , maar de computer beschouwt de eerste 10 cijfers als betekenisvol.

Het tiende cijfer wordt afgerond.

Als je invoert :

```
PRINT 1.23456787654321
```

zal APPLESOFT antwoorden met :

```
1.23456788
```

Bij optellingen en aftrekkingen kan het gebeuren , dat je waarden bereikt , die uitgaan boven de ' real precision ' grens !

Zo is het mogelijk , om het getal $1.7 \cdot 10^{38}$ te bereiken , zonder dat een foutmelding volgt.

Hier volgen nog enkele getalsvoorbeelden :

+1	wordt : 1
-1	wordt : -1
255	wordt : 255
-23.460	wordt : -23.46
$1 \cdot 10^{20}$	wordt : 1E+20

Type nu :

```
-12.3456789*10^10
```

Ben je erin gelopen ? Er gebeurde helemaal niets ?
Toch niet PRINT vergeten ? Het wendt heus wel !

Tracht te verklaren , waarom er het volgende te zien is :

1.2345679E+11

Als een getal wordt weergegeven , dan gelden de volgende regels.

- 1) als een getal negatief is , wordt het - teken afgedrukt.
- 2) als de absolute waarde van een getal een integer is tussen 0 en 999999999 , dan wordt hij geprint als integer.
- 3) als de absolute waarde van een getal groter is , of gelijk is aan 0.01 en kleiner dan 999999999.2 dan wordt dat getal zonder exponent afgedrukt.
- 4) als het getal niet tot groep 2 en 3 behoort , valt het onder de floating point notering.

2. Het Printformaat

Het tekstvenster van de APPLE-II is 40 karakters breed en 24 regels hoog. Het is mogelijk om dit venster kleiner te maken. Daarop komen we later terug.

Wanneer je een programmaregel invoert , zal er telkens een regel op het scherm open blijven. Dat is nodig , om de computer de mogelijkheid te geven tussen beide te komen. Als er iets mis gaat , antwoordt hij op de tussenregel.

Maar is de invoer langer , dan 1 regel , dan gaat de APPLE-II terdege op de eerstvolgende regel verder.

Probeer even :

```
PRINT "HET IS EEN HEEL GEWONE ZAAK , DAT DE APPLE ZO  
DOET"
```

Zie je - je moet het gewoon even weten. Om iets op het beeldscherm afgedrukt te krijgen , gebruiken we het commando PRINT (= druk af).

Een PRINT heeft de overgang naar een nieuwe regel tot gevolg , wanneer het niet wordt gevolgd door een ; of ,. Dat is van belang bij het samenvoegen van tekstregels. Wordt een PRINT commando afgesloten met ; dan wordt een volgend PRINT commando op dezelfde regel afgedrukt op de plaats , waar het eerste commando ophield !
Probeer het volgende programma :

```
10 PRINT "DIT IS";:PRINT "APPLE-II"  
20 PRINT "DIT IS"  
30 PRINT "EEN PROEF"  
40 PRINT "DIT IS"  
50 PRINT " APPLE-II"  
RUN
```

Op het beeldscherm verschijnt het volgende :

```
DIT ISAPPLE-II  
DIT IS  
EEN PROEF  
DIT IS APPLE-II
```

Houdt dus rekening met eventueel noodzakelijke spaties ! Je ziet welke functie de ; heeft. In regel 10 maakten we gebruik van het feit , dat meerdere statements op 1 regel mogen. Dit bespaart in belangrijke mate geheugenruimte. Ten hoogste kunnen er 239 karakters op 1 regel.

Een bezwaar van verschillende statements op 1 regel is de moeilijkheid eventuele fouten in zo'n lange regel later gemakkelijk te herstellen. De verschillende statement begrenzen we door : te typen.

Onderzoek :

```
10 PRINT," DAT ":PRINT "IS";:PRINT " TE GEK!"
```

Verklaar, waarom een RUN zo'n wonderlijk resultaat heeft!

Een PRINT commando , dat afgesloten wordt door een komma (,) heeft tot gevolg , dat een volgend PRINT commando 1 ' tab field ' verder wordt afgedrukt. Al eerder kwam dit ter sprake. Een regel is ingedeeld in 3 ' tab fields ' (= tabulatievelden). Erg gemakkelijk bij het maken van tabellen.

Laten we nu alle theorie met een praktisch voorbeeld aanvullen.

Hier is een programma met wat reistijden.

```
5 PRINT TAB(10) "*REISTIJDEN*"
10 PRINT"AFSTAND";:PRINT TAB(14)"SNELHEID";:PRINT
   TAB(30)"TIJD/UUR"
20 A=40:B=100
30 T=A/B
40 PRINT A,B,T
50 A=A+10
60 IF A=230 THEN 80
70 GOTO 30
80 END
```

LET OP ! !

Er zijn enige restricties aan het gebruik van de tabulatievelden :

De APPLE-II laat alleen het gebruik toe van het tweede tabulatie veld (pos. 17 tot 32) indien positie 16 van het eerste veld NIET beschreven is. Zo zal het derde tabulatie veld (pos. 33 tot 40) niet afgedrukt worden indien de posities 24 t / m 32 beschreven zijn. Hiermee dient rekening gehouden te worden als er een ' kop ' aan de tabellen gegeven dient te worden.

Vervang regel 10 in het bovenstaande programma eens door:

```
10 PRINT "AFSTAND",:PRINT"SNELHEID",:PRINT "TIJD/UUR"
```

Dit is wel even anders. De D op positie 24 is er de oorzaak van dat het derde ' tab field ' niet beschreven wordt.

3. Variabelen benoemen

Zoals uit het voorgaande , kleine oefenprogramma blijkt gebruiken we de letters A , B en T bij het uitvoeren van meervoudige berekeningen. De A voor de afstand , B voor de snelheid , die we de waarde 100 (k/u) gaven en tenslotte de T voor tijd. De A , B en T noemen wij ' variabelen '.

Regel 60 en 70 waren ' flow control ' commando's , die de loop van het programma beïnvloedden. Deze commando's komen later aan de orde.

Een ' variabele ' is niets meer , dan een geheugenlokatie binnen de APPLE-II computer , die aangeduid wordt met een letter.

De naam van een variabele moet beginnen met een letter en mag gevolgd worden door elk alfanumeriek karakter.

Een alfanumeriek is elke letter van A tot Z en elk cijfer van 0 tot 9.

Een naam van een variabele mag zelfs 238 karakters lang zijn !

DUS :

ABCDEFGHIJKLMNOPQRSTUVWXYZ.....

en dat 9 keer mag , maar bedenk dat APPLESOFT slechts de eerste 2 karakters gebruikt , om namen te onderscheiden.

M.a.w. zal een programma in de war raken , wanneer wij variabelen benoemen als GOUDVIS en GOUDK0ORTS. Alleen de letters G en O zijn van betekenis. Er is op deze regel (natuurlijk !) een uitzondering.

Wanneer wij een variabele benoemen , waarin een zgn. APPLESOFT gereserveerd woord zit , spoort APPLESOFT dit verboden en daarom ' illegale ' woord op en geeft een foutmelding.

Probeer :

```
10 SPOKEN=3
20 PRINT SPOKEN
30 END
RUN
```

en.' biep ' :

?SYNTAX ERROR IN 10

Je wijzigt de variabele SPOKEN in SPAKEN en geeft een RUN.

Het antwoord van de APPLE-II is nu :

3

Tracht nu in de lijst van gereserveerde woorden (zie aldaar) de oorzaak te vinden van de foutmelding.

LET OP !

Wijzig in regel 20 de variabele SPAKEN in SPEKEN en geef een RUN.

Het antwoord is :

3

Heb je nu door , wat er met het benoemen van variabelen aan de hand is ? Lees anders het voorafgaande opnieuw goed door.

Het benoemen van variabelen is van aanmerkelijk belang. Het is een middel , om geheugenruimte te sparen en de snelheid van de programma-uitvoering met een faktor 10 te verhogen.

Beginnen we bij een programma met het benoemen van de variabelen , dan kost het de APPLE-II later in het programma weinig tijd , de bijpassende waarden op te sporen.

Dit geldt ook voor programmaregels , die veelvuldig gebruikt moeten worden en waarnaar in het programma verwezen wordt. Neem deze regels zover mogelijk vooraan in je programma op.

Variabelen krijgen van de APPLE-II automatisch de waarde 0.

Je kunt de ingevoerde variabelen zelf van elke , andere waarde voorzien.

Slim is : $A=4$: $B=A+2$: $C=B-A$ i. p. v. $C=2$: $A=4$: $B=6$. Houdt er rekening mee dat de waarde van na het (=) teken wordt toegekend aan de waarde van voor het (=) teken.

De waarde van een variabele gaat verloren en de ruimte , die ervoor in het geheugen was vrijgemaakt , wordt anders bestemd , wanneer :

- a) wij een regel aan het programma toevoegen , of een regel eruit verwijderen.
- b) er een CLEAR commando wordt gegeven (zie aldaar).
- c) het RUN commando wordt gegeven.
- d) er NEW wordt getypt.

LET OP !

Dit betekent niet , dat de APPLE-II de door jouw benoemde waarde van een variabele op 0 gaat zetten. .. Er zit echt geen klein mannetje in de APPLE-II , die met het vlakgom rond gaat. ... Wel worden de waarden van afhankelijke variabelen op 0 gezet. Dus ergens in het programma wordt:

A + B = C C voorlopig 0

tot de loop van het programma er een andere waarde aan toekent , zo er nog een volgende loop komt.

De namen van zgn. integer variabelen moeten eindigen op %.

Bijvoorbeeld B% of C%.

De waarden van integer variabelen moeten liggen tussen -32767 en +32767 , zoals afspraak is voor gehele getallen.

Toch worden zij eerst omgerekend naar ' real precision' , voordat zij gebruikt worden in berekeningen. Dat is nodig om bij het terugrekenen naar een geheel getal afrondingsfouten te voorkomen.

Als een waarde wordt teruggeconverteerd naar integer (zo heet dat in computertaal) , dan wordt hij naar beneden afgerond.

Dus :

```
10 A%=.999
20 PRINT A%
RUN
0
```

Integer variabelen zijn niet toegestaan in FOR.... en DEF. .. statements.

Een reeks karakters wordt in APPLESOFT een ' literal ' genoemd. Je kunt dit ongeveer vertalen met het woord ' zin '.

Een literal tussen " " (aanhalingstekens) noemen we een ' string '. De string is van veel belang en biedt ons geweldige mogelijkheden.

```
10 PRINT " DIT IS EEN STRING "
RUN
```

Naast de alfanumerieke variabelen kennen we stringvariabelen.

Ook daaraan kunnen we een specifieke waarde toekennen. De namen van stringvariabelen moeten eindigen op een \$ teken.

Probeer :

```
NEW
10 A$="COMPUTER"
20 PRINT A$
RUN
COMPUTER
```

Nu we toch een ' waarde ' hebben toegekend aan een stringvariabele , kunnen we gelijk even vragen naar de lengte van de string :

```
20 PRINT A$,:PRINT LEN(A$),:PRINT LEN(" JA")
```

let nu goed op :

```
RUN
COMPUTER      8      3
```

Het LEN statement komt van LENGte. Hoe komt het , dat het beeldveld na de uitvoering van dit programma zo is ingedeeld ?

Waarom werd aan JA de lengte 3 toegekend?

Het aantal karakters van een string mag maximaal 255 bedragen.

De string mag niet langer zijn dan 255 leestekens , cijfers en spaties. Probeer zo'n zin te maken en verbaas je over de ontzettende lengte , die bereikt kan worden.

Een string die geen karakters bevat , heet een ' nul-string '.

Er worden geen karakters van afgedrukt (hoe kan het , zou je zeggen) en de cursor slaat hem a. h. w. over bij het maken van kolommen.

Type :

```
30 PRINT LEN(Q$);Q$,3
RUN 30
0      3
```

Vergelijk de schrijfwijze van de verschillende instructies in dit programma. Er zijn klaarblijkelijk verschillende mogelijkheden , die tot hetzelfde resultaat (tekstverdeling) leiden. Welke zal het zuinigst omspringen met de geheugenruimte ?

Samenvoeging van strings is mogelijk. Je maakt gewoon een optelsom van strings door er een + teken tussen te plaatsen.

Beproef het volgende :

```
NEW
10 A$="HET GAAT"
20 B$=A$+" GOED MET DE APPLE-II "
30 PRINT B$
RUN
HET GAAT GOED MET DE APPLE-II
```

Denk altijd om de noodzakelijke spaties , die overigens terdege aangemerkt worden als karakters. Je ziet ze niet - ze zijn er toch en tellen mee.

Interessant is , dat A , A\$ en A% aangemerkt worden als 3 verschillende variabelen. Zij mogen tegelijk in een programma voorkomen.

Tijd voor een kleine samenvatting :

```
10 A$="HET GAAT"
20 B$=A$+" GOED MET MIJ"
30 PRINT B$
40 A%=3.8765
50 A=5.6780
60 PRINT LEN(B$),A%,A
RUN
HET GAAT GOED MET MIJ
21      3      5.678
```

Het zal je duidelijk zijn , dat variabelen en strings een belangrijke rol bij het programmeren spelen.

Wanneer een onderwerp niet duidelijk is , lees dit dan beslist nog een keer over!

Let op de funktie van de : de , en de ;.

4. Logische en rekenkundige regels

Misschien herinner je je de beroemde rekenkundige ezelsbrug :

' Meneer Van Dale Wacht Op Antwoord '.

Eerst machtsverheffen , daarna vermenigvuldigen , delen , worteltrekken , optellen en aftrekken. Wanneer je en ingewikkelde rekensom moest uitrekenen , hield je met deze regel rekening.

Anders zou de som als $2 + 3 * 6$ voor jou 30 tot resultaat hebben , i. p. v. 20. Bij de APPLE-II Personal Computer is dit niet anders.

Er bestaan voor de APPLE-II een aantal 'voorrangsregels' die wij hierna zullen opsommen. Omdat de expressies (zeg : getallen) van links naar rechts worden uitgewerkt , zullen operatoren met eenzelfde ' voorrang ' (prioriteit) in volgorde van hun verschijning worden afgewerkt. Optellen , delen etc. noemt men in de wiskunde een ' operatie ' en het teken , dat bij zo'n operatie wordt gebruikt heet de operator (bijv. + of -). Wanneer je een expressie tussen () schrijft , gelden binnen de haakjes de normale voorrangsregels , echter wordt hetgeen tussen de (en) staat eerst uitgewerkt. Daarna wordt de uitkomst met eventuele expressies buiten de haakjes bewerkt.

```
PRINT (2+3)*6
RUN
30
```

Er zijn 3 categorieën operaties te onderscheiden :

- a) Wiskundige operaties
- b) Relationele operaties
- c) Logische operaties

Ad a) tekens zijn + - * / =

Ad b) tekens zijn = < <= > >= <>

Ad c) tekens zijn NOT OR AND

De voorrangsregeling van deze operatoren is als volgt :

1. () Formules tussen haakjes eerst uitwerken
2. NOT , + - De + en - als teken voor exponent ; de ontkenning.
3. ^ Machtsverheffen
4. /* Delen , vermenigvuldigen in volgorde van langskomen.
5. +- Optellen , aftrekken in volgorde van langskomen.
6. De relatie-operatoren in volgorde van langskomen.
7. De logische operatoren , eerst AND , daarna OR.

De logische operatoren kunnen worden gebruikt voor Boolese Vergelijkingen , waarbij de logica getoetst wordt aan de rekenkunde. We kunnen de APPLE-II daarmee ' de waarheid ' laten spreken. Wat waar is , wordt 1 - wat onwaar is 0.

```
PRINT NOT 1,NOT 0
RUN
0      1
```

De APPLE-II vindt , dat wat niet waar is , onwaar is en wat niet onwaar is waar ! Echt iets , om op een regenachtige dag over na te denken.

5. Nog iets over commando ' s

Zoals je hebt gezien , mogen een aantal commando ' s in APPLESOFT worden gebruikt in de zgn. ' immediate mode ' (spreek uit : immidie-ut mood). Deze commando ' s werden zonder regelnummer ingetypt en direkt na RETURN uitgevoerd.

De ' immediate mode ' betekent ' onmiddellijke werking '.

In tegenstelling hiermee maakten we enkele programma ' s in de ' deferred mode ' (spreek uit : diefurd mode) , waarbij we van regelnummers gebruik maakten.

' Deferred mode ' wil zoveel zeggen als ' vertraagde werking '.

De regelnummers mogen gaan van 0 tot 63999.

Het is een gewoonte , om tussen de regelnummers wat afstand te bewaren voor latere tussenvoegingen en veranderingen.

Vandaar dat wij van regel 10 naar 20 en van 20 naar 30 over gingen. Dan hebben we wat ruimte voor extra regelnummers ertussen.

In de deferred mode wordt de return-toets gebruikt om een programmaregel in het geheugen op te slaan. Door het commando RUN worden de instructies van het programma uitgevoerd , te beginnen bij de laagste regel. De APPLE-II ' leest ' dus het programma draaiboek punt voor punt door en voert uit , wat er van hem wordt verlangd. Het uitvoeren van een programma noemt men ook wel programma - ' executie '. Dit nogal griezelige woord duikt vaak in vakliteratuur op .

In de hierna volgende hoofdstukken zal met de afkorting 'imm ' en ' def ' worden aangegeven in welke ' mode ' er commando ' s mogen worden gegeven.

We geven een zo bondig mogelijk overzicht van de vele commando ' s voor de APPLE-II Personal Computers.

HOOFDSTUK 4 ENKELE OVERZICHTEN

1 . Commando samenvatting

commando	uitspraak	
ABS		MID\$
ASC		NEW
ATN		NEXT
CALL	: kol	NORMAL
CHR\$		NOTRACE
CLEAR	: klier	ON. GOTO
COLOR		ON. GOSUB
CONT		ONERR. GOTO
COS		PDL
CTRL-C	: kontrol	PEEK
CTRL-X		PLOT
DATA		POKE
DEF FN		POP
DEL		POS
DIM		PRINT
DRAW. .AT	: drow et	PR #
END		READ
ESCAPE A,B,C,D	: eskeep	RECALL
EXP		REM
FLASH	: flesj	REPT
FOR. .TO. .STEP. .		RESTORE
FRE		RESUME
GET		RETURN
GOSUB		RIGHT\$
GOTO		RND
GR		ROT
HCOLOR		RUN
HGR	: haai-greffiks	SAVE
HGR2		SCALE
HIMEM	: haai-mem	SCRN
HLIN. .AT		SGN
HOME	: hoom	SHLOAD
HPLOT		SIN
HTAB		SPC
IF. .THEN		SPEED
INPUT		SQR
INT		STOP
INVERSE	: invurz	STORE
IN #		STR\$
LEFT\$		TAB
LEN		TAN
LET		TEXT
LIST		TRACE
LOAD	: lood	USR
LOG		VAL
LOMEM		
		: noo-trees
		: piek
		: pook
		: ried
		: riekol
		: ris-toor
		: riesjoehm
		: raajt
		: seef
		: skeel
		: spied
		: stohr

Alfabetisch overzicht gereserveerde woorden

Token	Reserved Word	Token	Reserved Word	Token	Reserved Word
128	END	164	LOMEM:	200	+
129	FOR	165	ONERR	201	-
130	NEXT	166	RESUME	202	*
131	DATA	167	RECALL	203	/
132	INPUT	168	STORE	204	^
133	DEL	169	SPEED=	205	AND
134	DIM	170	LET	206	OR
135	READ	171	GOTO	207	>
136	GR	172	RUN	208	=
137	TEXT	173	IF	209	<
138	PR#	174	RESTORE	210	SGN
139	IN#	175	&	211	INT
140	CALL	176	GOSUB	212	ABS
141	PLOT	177	RETURN	213	USR
142	HLIN	178	REM	214	FRE
143	VLIN	179	STOP	215	SCRN(
144	HGR2	180	ON	216	PDL
145	HGR	181	WAIT	217	POS
146	HCOLOR=	182	LOAD	218	SQR
147	HPlot	183	SAVE	219	RND
148	DRAW	184	DEF	220	LOG
149	XDRAW	185	POKE	221	EXP
150	HTAB	186	PRINT	222	COS
151	HOME	187	CONT	223	SIN
152	ROT=	188	LIST	224	TAN
153	SCALE=	189	CLEAR	225	ATN
154	SHLOAD	190	GET	226	PEEK
155	TRACE	191	NEW	227	LEN
156	NOTRACE	192	TAB(228	STR\$
157	NORMAL	193	TO	229	VAL
158	INVERSE	194	FN	230	ASC
159	FLASH	195	SPC(231	CHR\$
160	COLOR=	196	THEN	232	LEFT\$
161	POP	197	AT	233	RIGHT\$
162	VTAB	198	NOT	234	MID\$
163	HIMEM:	199	STEP		

2. Iets over Integer BASIC

Zoals in de geheugenmap van APPLESOFT is te zien, is er een tweede BASIC interpreter in de APPLE - II aanwezig. Dit is een integer BASIC - Interpreter. Constanten en waarden van variabelen in integer BASIC moeten gehele getallen zijn en liggen tussen de waarden - 32767 en + 32767.

Vanuit Monitor kan Integer BASIC worden opgestart door :

RESET , C081 , RETURN , CTRL-B , RETURN

Voor systemen die uitgerust zijn met een Autostart ROM , moet je i.p.v RESET , CALL - 151 intypen.

Het ' prompt ' karakter van Integer BASIC is >.

Er zijn behalve het getalsysteem een aantal verschillen tussen APPLESOFT en INTEGER BASIC :

De volgende commando ' s bestaan niet in Integer BASIC :

ATN					
CHR\$	COS				
DATA	DEF FN	DRAW			
EXP					
FLASH	FN	FRE			
GET					
HCOLOR	HGR	HGR2	HIMEM:	HOME	HPLLOT
INT	INVERSE				
LEFT\$	LG	LOMEM:			
MID\$					
NORMAL					
ON.GOSUB	ON.GOTO	ONERR.GOTO			
POS					
READ	RECALL	RESTORE	RESUME	RIGHT\$	ROT
SCALE	SHLOAD	SIN	SPC	SPEED	SQR
STOP	STORE				
STR\$					
TAN					
USR					
VAL					
WAIT					
XDRAW					

De volgende commando ' s komen in Integer BASIC voor ,
maar niet in APPLESOFT :

AUTO rnum [,step]

Nummert BASIC regels automatisch vanaf rnum.
Als step niet gedefinieerd is , is de stapgrootte 10 ,
anders / step /.

DSP var

Monitor functie.

Elke keer , als var van waarde verandert , wordt
gedisplayed : var = nieuwe waarde.

RUN reset de DSP functie , zodat het alleen maar zin
heeft de DSP als ' DEFERRED ' commando te gebruiken.

MAN

Zet de AUTO regelnummering af.

MOD

Rest na deling. Modulus. $12 \text{ MOD } 5 = 2$

De volgende commando ' s hebben verschillende namen in
beide BASIC versies :

INTEGER BASIC

APPLESOFT

CLR

CLEAR

CON

CONT

TAB

HTAB (APPLESOFT HEEFT OOK TAB)

GOTO X*10+100

ON X GOTO 100,110,120

GOSUB X 100+100

ON X GOSUB 1000,1100,1200

CALL -936

HOME of CALL -936

POKE 50,127

INVERSE

POKE 50,255

NORMAL

X

X% (% geeft integer variabele aan)

#

<> of ><

Andere verschillen

Bij Integer BASIC wordt de juistheid van statements bij het invoeren gecontroleerd ; bij APPLESOFT pas bij het executeren.

GOTO en GOSUB moeten in APPLESOFT gevolgd worden door een regelnummer , bij Integer BASIC mag dat ook een expressie zijn.

In APPLESOFT zijn alleen de eerste twee karakters van een naam significant , bij Integer BASIC alle karakters van een naam.

Bij Integer BASIC moeten zowel array ' s als strings geDIMensioneerd worden , bij APPLESOFT alleen array ' s. Array ' s in Integer BASIC mogen maar een dimensie hebben, in APPLESOFT verschillende.

Array elementen worden in APPLESOFT 0 gemaakt door RUN , CLEAR en RESET , CTRL - B , RETURN. Bij Integer BASIC moet de gebruiker array elementen zelf 0 maken. In APPLESOFT mogen POKE ' s , PEEKS en CALL ' s adressen groter dan 32767 voorkomen. In Integer BASIC moet zo 'n adres aangeroepen worden met de ' two ' s complement ' negatieve waarde.

END in een programma , dat in de hoogst voorkomende regel stopt , is niet verplicht in APPLESOFT , wel in Integer BASIC.

NEXT moet (!) bij Integer BASIC gevolgd worden door een naam van een variabele , in APPLESOFT hoeft geen naam voor te komen.

Bij Integer BASIC wordt bij het INPUT statement een ? afgedrukt , als een arithmetische waarde wordt verwacht , ongeacht het feit of in het INPUT statement een tekststring was gedefinieerd.

Het vraagteken wordt niet afgedrukt , als een stringconstante wordt verwacht.

Bij de APPLESOFT INPUT wordt het vraagteken afgedrukt als geen tekststring was gespecificeerd , maar weggelaten als dat wel het geval was.

In APPLESOFT wordt de optionele string afgesloten met ;. Bij Integer BASIC met ,.

Foutmeldingen van Integer BASIC
EQUIVALENT

XXX SYNTAX ERROR
XXX TOO LONG ERROR
XXX 32767 ERR
XXX 255 ERR
STOPPED AT YYYY
XXX BAD BRANCH ERR
STATEMENT

XXX BAD RETURN ERR
GOSUB
XXX BAD NEXT ERR
ERROR
XXX 16 GOSUBS ERROR
ERROR
XXX 16 FORS ERR
ERROR
XXX NO END ERROR
XXX MEM FULL
ERROR
XXX DIM ERR
ERROR
XXX RANGE ERR
ERROR
XXX STR OVFL ERROR
XXX STRING ERR
RETYPE LINE

APPLESOFT

?SYNTAX ERROR
?STRING 100 LONG
?OVERFLOW ERROR
?ILLEGAL QUANTITY

?UNDEF ' D

ERROR
?RETURN WITHOUT

?NEXT WITHOU I FOR

?OUT OF MEMORY

?OUT OF MEMORY

?OUT OF MEMORY

?REDIM ' D ARRAY

?BAD SUBSCRIPT

?STRING 100 LONG
?SYNTAX ERROR
?REENIER

3. ASCII - codes

ASCII Character Codes

ASCII Code	Display Screen Character	Keystroke	ASCII Code	Display Screen Character	Keystroke
0		CTRL-@	48	0	O
1		CTRL-A	49	1	1
2		CTRL-B	50	2	2
3		CTRL-C	51	3	3
4		CTRL-D	52	4	4
5		CTRL-E	53	5	5
6		CTRL-F	54	6	6
7	(bell)	CTRL-G	55	7	7
8	(backspace)	CTRL-H or ←	56	8	8
9		CTRL-I	57	9	9
10	(linefeed)	CTRL-J	58	:	:
11		CTRL-K	59	;	;
12		CTRL-L	60	<	<
13	(carriage return)	CTRL-M	61	=	=
14		CTRL-N	62	>	>
15		CTRL-O	63	?	?
16		CTRL-P	64	@	@
17		CTRL-Q	65	A	A
18		CTRL-R	66	B	B
19		CTRL-S	67	C	C
20		CTRL-T	68	D	D
21	(forward space)	CTRL-U or →	69	E	E
22		CTRL-V	70	F	F
23		CTRL-W	71	G	G
24	(cancel line)	CTRL-X	72	H	H
25		CTRL-Y	73	I	I
26		CTRL-Z	74	J	J
27		Esc	75	K	K
28		n.a.	76	L	L
29		CTRL-SHIFT-M	77	M	M
30		CTRL- ^	78	N	N
31		n.a.	79	O	O
32	space	space bar	80	P	P
33	!	!	81	Q	Q
34	"	"	82	R	R
35	#	#	83	S	S
36	\$	\$	84	T	T
37	%	%	85	U	U
38	&	&	86	V	V
39	'	'	87	W	W
40	((88	X	X
41))	89	Y	Y
42	*	*	90	Z	Z
43	+	+	91	[n.a.
44	,	,	92	\	n.a.
45	-	-	93]	SHIFT-M
46	.	.	94	^	^
47	/	/	95		n.a.

n.a. = not available on the Apple II keyboard.

ASCII codes van 96 tot 255 genereren karakters , die een duplicaat zijn van de hiervoor vermelde.
 Ofschoon CHR\$(65) en CHR\$(193) beide een A teruggeven , herkent APPLE - II de beide karakters niet als hetzelfde karakter !!!


```

*****
*                                     *
*  APPLE-II FLOATING                *
*  POINT ROUTINES                   *
*                                     *
*  COPYRIGHT 1977 BY                *
*  APPLE COMPUTER INC.              *
*                                     *
*  ALL RIGHTS RESERVED              *
*                                     *
*      S. WOZNAK                    *
*                                     *
*****

```

4. TITLE "FLOATING POINT ROUTINES"

	SIGN	EPZ	\$F3		
	X2	EPZ	\$F4		
	M2	EPZ	\$F5		
	X1	EPZ	\$F8		
	M1	EPZ	\$F9		
	E	EPZ	\$FC		
	OVLOC	EDU	\$3F5		
		ORG	\$F425		
F425: 18	ADD	CLC			CLEAR CARRY.
F426: A2 02		IDX	#S2		INDEX FOR 3-BYTE ADD.
F428: B5 F9	ADD1	LDA	M1,X		
F42A: 75 F5		ADC	M2,X		ADD A BYTE OF MANT2 TO MANT1.
F42C: 95 F9		STA	M1,X		
F42E: CA		DEY			INDEX TO NEXT MORE SIGNIF. BYTE.
F42F: 10 F7		BPL	ADD1		LOOP UNTIL DONE.
F431: 60		RTS			RETURN
F432: 06 F3	MD1	ASL	SIGN		CLEAR LSB OF SIGN.
F434: 20 37 F4		JSR	ABSWAP		ABS VAL OF M1, THEN SWAP WITH M2
F437: 24 F9	AESWAP	BIT	M1		MANT1 NEGATIVE?
F439: 10 05		BPL	ABSWAP1		NO, SWAP WITH MANT2 AND RETURN.
F43B: 20 A4 F4		JSR	FCOMPL		YES, COMPLEMENT IT.
F43E: E6 F3		INC	SIGN		INC SIGN, COMPLEMENTING LSB.
F440: 38	ABSWAP1	SEC			SET CARRY FOR RETURN TO MUL/DIV.
F441: A2 04	SWAP	LDX	#S4		INDEX FOR 4-BYTE SWAP.
F443: 94 FB	SWAP1	STY	E-1,X		
F445: B5 F7		LDA	X1-1,X		SWAP A BYTE OF EXP/MANT1 WITH
F447: 94 F3		LDY	X2-1,X		EXP/MANT2 AND LEAVE A COPY OF
F449: 94 F7		STY	X1-1,X		MANT1 IN E (3 BYTES). E+3 USED
F44B: 95 F3		STA	X2-1,X		
F44D: CA		DEY			ADVANCE INDEX TO NEXT BYTE.
F44E: D0 F3		BNE	SWAP1		LOOP UNTIL DONE.
F450: 60		RTS			RETURN
F451: A9 B5	FLOAT	LDA	#SFE		INIT EXPL TO 14,
F453: B5 F6		STA	X1		THEN NORMALIZE TO FLOAT.
F455: A5 F9	NORM1	LDA	M1		HIGH-ORDER MANT1 BYTE.
F457: C9 C0		CMF	#SC0		UPPER TWO BITS UNEQUAL?
F459: 30 0C		RMI	PTS1		YES, RETURN WITH MANT1 NORMALIZED
F45B: C6 F8		DEC	X1		DECREMENT EXPL.
F45D: 06 FF		ASL	M1+2		
F45F: 26 FA		ROL	M1+1		SHIFT MANT1 (3 BYTES) LEFT.
F461: 26 F9		ROL	M1		
F463: A5 F8	NORM	LDA	X1		EXPL ZERO?
F465: D0 EE		BNE	NORM1		NO, CONTINUE NORMALIZING.
F467: 60		RTS			RETURN.
F468: 20 A4 F4	FSUR	JSR	FCOMPL		CMPL MANT1, CLEARS CARRY UNLESS 0
F46B: 20 7B F4	SWPALGN	JSR	ALGNSWP		RIGHT SHIFT MANT1 OR SWAP WITH
F46E: A5 F4	FADD	LDA	X2		
F470: C5 F8		CMF	X1		COMPARE EXPL WITH EXP2.
F472: D0 F7		BNE	SWPALGN		IF #, SWAP ADDENDS OR ALIGN MANTS.
F474: 20 25 F4		JSR	ADD		ADD ALIGNED MANTISSAS.
F477: 50 EA	ADDEND	BVC	NORM		NO OVERFLOW, NORMALIZE RESULT.
F479: 70 05		BVS	RTLOG		OV: SHIFT M1 RIGHT, CARRY INTO SIGN
F47B: 90 C4	ALGNSWP	BCC	SWAP		SWAP IF CARRY CLEAR,
			FALSE SHIFT RIGHT ARITH.		
F47D: A5 F9	RTAP	LDA	M1		SIGN OF MANT1 INTO CARRY FOR
F47F: 0A		ASL	A		RIGHT ARITH SHIFT.
F480: E6 F8	RTLOC	INC	X1		INCR X1 TO ADJUST FOR RIGHT SHIFT
F482: F0 75		PEC	OVFL		EXPL OUT OF RANGE.
F484: A2 FA	RTLOG1	LDX	#SFA		INDEX FOR 6-BYTE RIGHT SHIFT.
F486: 76 FF	ROP1	ROR	E+3,X		
F488: E8		INX			NEXT BYTE OF SHIFT.
F489: D0 FB		BNE	ROP1		LOOP UNTIL DONE.
F48B: 60		RTS			RETURN.
F48C: 20 32 F4	MUL	JSR	MD1		ABS VAL OF MANT1, MANT2.
F48F: 65 F8		ADC	X1		ADD EXPL TO EXP2 FOR PRODUCT EXP
F491: 20 E2 F4		JSR	MD2		CHECK PROD. EXP AND PREP. FOR MUL
F494: 18		CLC			CLEAR CARRY FOR FIRST BIT.
F495: 20 84 F4	MUL1	JSR	RTLOG1		M1 AND E RIGHT (PROD AND MPLIEP)
F498: 90 03		BCC	MUL2		IF CARRY CLEAR, SKIP PARTIAL PROD
F49A: 20 25 F4		JSR	ADD		ADD MULTIPLICAND TO PRODUCT.
F49D: B8	MUL2	DEY			NEXT MUL ITERATION.
F49E: 10 F5		BPL	MUL1		LOOP UNTIL DONE.

F4A0: 46 F3	NDEND	LSR	SIGN	TEST SIGN LSR.
F4A2: 90 2F	NORMX	RCC	NORM	IF EVEN,NORMALIZE PROD,ELSE COMP
F4A4: 38	FCOMPL	SEC		SET CARRY FOR SUBTRACT.
F4A5: A2 03		LDX	#S3	INDEX FOR 3-BYTE SUBTRACT.
F4A7: A9 00	CO4PL1	LDA	#S0	CLEAR A.
F4A9: F5 F8		SBC	X1,X	SUBTRACT BYTE OF EXP1.
F4AB: 95 F8		STA	X1,X	RESTORE IT.
F4AD: CA		DEX		NEXT MORE SIGNIFICANT BYTE.
F4AE: D0 F7		BNE	COMPL1	LOOP UNTIL DONE.
F4B0: F0 C5		BEQ	ADDEND	NORMALIZE (OR SHIFT RT IF OVFL).
F4B2: 20 32 F4	FDIV	JSR	MD1	TAKE ABS VAL OF MANT1, MANT2.
F4B5: E5 F8		SRC	X1	SUBTRACT EXP1 FROM EXP2.
F4B7: 20 E2 F4		JSR	MD2	SAVE AS QUOTIENT EXP.
F4BA: 38	DIV1	SEC		SET CARRY FOR SUBTRACT.
F4BB: A2 02		LDX	#S2	INDEX FOR 3-BYTE SUBTRACTION.
F4BD: B5 F5	DIV2	LDA	M2,X	
F4BF: F5 FC		SBC	E,X	SUBTRACT A BYTE OF E FROM MANT2.
F4C1: 48		PHA		SAVE ON STACK.
F4C2: CA		DEX		NEXT MORE SIGNIFICANT BYTE.
F4C3: 10 F8		DPL	DIV2	LOOP UNTIL DONE.
F4C5: A2 FD		LDX	#SFD	INDEX FOR 3-BYTE CONDITIONAL MOVE
F4C7: 68	DIV3	PLA		PULL BYTE OF DIFFERENCE OFF STACK
F4C8: 90 02		BCC	DIV4	IF M2<E THEN DON'T RESTORE M2.
F4CA: 95 F8		STA	M2+3,X	
F4CC: E8	DIV4	INX		NEXT LESS SIGNIFICANT BYTE.
F4CD: D0 F8		BNE	DIV3	LOOP UNTIL DONE.
F4CF: 26 FB		ROL	M1+2	
F4D1: 26 FA		ROL	M1+1	ROLL QUOTIENT LEFT,CARRY INTO LSR
F4D3: 26 F9		ROL	M1	
F4D5: 06 F7		ASL	M2+2	
F4D7: 26 F6		ROL	M2+1	SHIFT DIVIDEND LEFT.
F4D9: 26 F5		ROL	M2	
F4DB: B0 1C		BCS	OVFL	OVFL IS DUE TO UNNORMED DIVISOR
F4DD: 88		DEY		NEXT DIVIDE ITERATION.
F4DE: D0 DA		BNE	DIV1	LOOP UNTIL DONE 23 ITERATIONS.
F4E0: F0 BE		BEQ	MDEND	NORM. QUOTIENT AND CORRECT SIGN.
F4E2: 86 FB	MD2	STX	M1+2	
F4E4: 86 FA		STX	M1+1	CLEAR MANT1 (3 BYTES) FOR MUL/DIV.
F4E6: 86 F9		STX	M1	
F4E8: B0 0D		BCS	OVCHK	IF CALC. SET CARRY,CHECK FOR OVFL
F4EA: 30 04		BMI	MD3	IF NEG THEN NO UNDERFLOW.
F4EC: 68		PLA		POP ONE RETURN LEVEL.
F4ED: 68		PLA		
F4EE: 90 B2		RCC	NORMX	CLEAR X1 AND RETURN.
F4F0: 49 80	MD3	EOR	#S80	COMPLEMENT SIGN BIT OF EXPONENT.
F4F2: 85 F8		STA	X1	STORE IT.
F4F4: A0 17		LDY	#S17	COUNT 24 MUL/23 DIV ITERATIONS
F4F6: 60		PTS		RETURN.
F4F7: 10 F7	OVCHK	BPL	MD3	IF POSITIVE EXP THEN NO OVFL.
F4F9: 4C F5 03	OVFL	JMP	OVLOC	
		ORG	\$F63D	
F63D: 20 7D F4	FIX1	JSR	RTAR	
F640: A5 F8	FIX	LDA	X1	
F642: 10 13		BPL	UNDFL	
F644: C9 8E		CMP	#S8E	
F646: D0 F5		DNF	FIX1	
F648: 24 F9		RIT	M1	
F64A: 10 0A		BPL	FIXPTS	
F64C: A5 FB		LDA	M1+2	
F64E: F0 06		BEO	FIXRTS	
F650: E6 FA		INC	M1+1	
F652: D0 02		BNE	FIXRTS	
F654: E6 F9		INC	M1	
F656: 60	FIXRTS	PTS		
F657: A9 00	UNDFL	LDA	#S0	
F659: 85 F9		STA	M1	
F65B: 85 FA		STA	M1+1	
F65D: 60		RTS		

HOOFDSTUK 5 DE BESCHIKBARE COMMANDO'S

1. Systeem commando's

LOAD imm & def
SAVE imm & def

load
save

Deze commando's laden een programma van de cassetteband in het geheugen of schrijven een programma naar de cassette. Je moet de cassetterecorder aan hebben (weergave of opname) als deze commando's gegeven worden. LOAD en SAVE controleren niet of de cassetterecorder aan staat , of wel aanwezig is. Beide commando's laten een signaaltje horen aan het begin en het einde van hun werking.

De cursor verdwijnt van het scherm , als een van deze commando's actief is en verschijnt weer als de LOAD , of SAVE klaar is.

Uitvoering van het programma wordt gestopt na een LOAD commando.

Bij een SAVE wordt het programma voortgezet na het wegschrijven van de gegevens.

Zie ook het hardware-gedeelte van dit boek (interfacing).

Alleen een RESET kan een LOAD of SAVE onderbreken.

NEW imm & def

new

Dit commando verwijdert programma's en variabelen uit het geheugen ! (In feite worden ze ontoegankelijk).

RUN imm & def

run rnum

Dit commando zet alle variabelen op 0 en begint de uitvoering van het regelnummer , dat eventueel gespecificeerd is. Als geen regelnummer is opgegeven , wordt begonnen met de regel met het laagste regelnummer.

Als dit commando in 'def' mode met een regelnummer wordt gebruikt en het regelnummer bestaat niet, of is negatief, dan wordt de foutmelding :

?UNDEF'D STATEMENT ERROR

gegeven.

Als het regelnummer groter is dan 63999 , dan verschijnt:

?SYNTAX ERROR

op het scherm.

De regel , waar de fout optrad , wordt niet vermeld.

In de immediate mode worden de foutmeldingen echter :

?UNDEF ' D STATEMENT ERROR IN XXXX

en

?SYNTAX ERROR IN XXXX

Waar XXXX een regelnummer aangeeft.

STOP imm & def

stop

STOP stopt programma - executie en geeft de controle over het systeem aan je terug.

De melding :

BREAK IN XXX

vertelt dat , XXX het regelnummer is , waarin het STOP commando voorkwam.

END imm & def

end

Dit programma stopt de programma - executie , en geeft het toetsenbord aan je terug.

CONT imm & def

cont

Als de uitvoering van een programma is afgebroken met een END , met een STOP , of met CTRL - C kan programma - executie weer voortgezet worden met een CONTInue

Het programma wordt dan hervat bij de volgende instructie, niet altijd bij het volgende regelnummer. Als er geen onderbroken programma is , heeft CONT geen effect. Variabelen worden niet op 0 gezet.

Als een INPUT instructie is onderbroken met een CTRL - C, resulteert een CONT commando in :

?SYNTAX ERROR IN XXXX

waar XXXX het regelnummer is , waar het INPUT statement voorkomt.

Een CONT resulteert in de foutmelding :

?CAN'T CONTINUE ERROR

wanneer na de onderbreking van het programma :

1. Een programmaregel wordt gewijzigd , of verwijderd.
2. Een commando wordt uitgevoerd , dat resulteert in een foutmelding.

Programmavariabelen mogen gewijzigd worden door het gebruik van immediate instructies. Er mag dan echter geen foutmelding optreden.

TRACE imm & def

trace

TRACE brengt een zoekprogramma op gang , waarbij elk regelnummer wordt afgedrukt op het moment , dat aan de uitvoering ervan wordt begonnen.

Als het programma ook PRINT statements bevat , is het mogelijk , dat TRACE deze informatie op het scherm overschrijft. Vooral bij het zoeken naar fouten is TRACE nuttig.

De foutzoek-werking noemt men 'DEBUG - mode '.

NOTRACE imm & def

notrace

Reset de TRACE mode.
Is noodzakelijk na TRACE.

PEEK imm & def

peek (aexpr)

Deze instructie geeft (DECIMAAL) de inhoud van geheugenplaats / aexpr /.

In hoofdstuk 5 zijn verschillende voorbeelden van het gebruik van PEEK opgenomen.

POKE imm & def

poke aexpr1 , aexpr2

Dit commando schrijft op geheugenlokatie aangegeven door / aexpr1 / het binaire equivalent van / aexpr2 /. De waarde van aexpr1 moet liggen tussen -65535 en 65535 , terwijl / aexpr2 / moet liggen tussen 0 en 255. Anders wordt de foutmelding :

?ILLEGAL QUANTITY ERROR

afgedrukt.

Dit commando heeft natuurlijk alleen effect , als het gespecificeerde adres in het systeem aanwezig is (RAM , output).

In het algemeen betekent dit , dat / aexpr1 / niet groter mag zijn dan 'memtop '. Memtop is de hoogste geheugenlokatie , die in het systeem voorkomt. Bij een 16K systeem is dat 16384 , bij een 32K systeem 32768 en bij een 48K systeem 49125.

Een poging om een PROM - lokatie te beschrijven heeft natuurlijk nimmer effect. Een aantal geheugenadressen bevat belangrijke informatie , om het systeem correct te laten functioneren. Een willekeurige POKE in deze lokaties kan later problemen veroorzaken.

WAIT imm & def

wait aexpr1 , aexpr2 ,aexpr3.

WAIT geeft je de mogelijkheid , om een konditionele pauze in een programma in te lassen. Alleen een RESET kan een WAIT voortijdig afbreken.

/ aexpr1 / is de geheugenlokatie , waarop de test wordt uitgevoerd.

Voor de waarde van aexpr1 geldt hetzelfde als bij aexpr1 van het POKE commando. / aexpr1 / en / aexpr2 / moeten liggen tussen 0 en 255.

Als alleen aexpr1 en aexpr2 zijn gespecificeerd , wordt elk bit van de aexpr1 gespecificeerde geheugenlokatie geAND met het binaire equivalent van aexpr2. Als dit resultaat 0 geeft , wordt de test opnieuw uitgevoerd. Als de test een resultaat ongelijk 0 oplevert , dan wordt het programma voortgezet.

Als aexpr2 en aexpr3 beide zijn gegeven , wordt eerst elk bit van de lokatie / aexpr1 / ge-EXCLUSIVE ORed met het binaire equivalent van aexpr3 en vervolgens wordt het resultaat geAND met / aexpr2 /. Een en ander heeft tot gevolg , dat een test wordt uitgevoerd op in / aexpr2 / gespecificeerde bits. Is een betreffend bit niet 1 in /aexpr3 / , dan wordt de test herhaald , tot een van deze bits 1 wordt. Komen in / aexpr2 / en / aexpr3 / bits voor die beide 1 zijn , dan wordt de test herhaald tot een van deze bits laag wordt.

WAIT is zeer geschikt om een programma te synchroniseren met een I/O - device.

Het biedt de mogelijkheid te synchroniseren op H00G - LAAG en op LAAG - H00G overgangen.

Voorbeelden :

WAIT aexpr1,255,0	betekent dat gewacht wordt tot tenminste een bit van de lokatie / aexpr1 / 1 is.
WAIT aexpr1,255	idem
WAIT aexpr1,255,255	Pauze tot minstens 1 van de 8 bits van het geheugen (of I/O) lokatie 0 is.
WAIT aexpr1,1,1	wacht tot het minstsignificante bit 0 is , ongeacht de andere bits
WAIT aexpr1,3,2	wacht tot het minstsignificante bit 1 is , of het volgende bit 0 is , of beide.

CALL imm & def

call aexpr

Start executie van een machinetaal subroutine , die begint op adres / aexpr /.

HIMEM imm & def

himem:aexpr

Deze instructie zet het hoogste voor BASIC programma's beschikbare adres op / aexpr /.

Het wordt gebruikt om , een geheugengebied waar gegevens, grafische informatie , of machinetaal programma's staan te beschermen.

Normaal wordt HIMEM automatisch ingesteld op het hoogste in het systeem beschikbare RAM adres. Dit gebeurt bij het opstarten van APPLESOFT (RESET , CTRL - B , RETURN).

De waarde van HIMEM staat op de geheugenplaatsen 115 en 116 (decimaal).

Om dit zichtbaar te maken , gebruik :

```
PRINT PEEK(116)*256+PEEK(115)
```

Als HIMEM wordt gezet op een adres lager dan LOMEM , of er is niet meer voldoende geheugenruimte voor BASIC programma's aanwezig , dan wordt de volgende foutmelding gegeven :

?OUT OF MEMORY ERROR

HIMEM wordt gereset door RESET , CTRL - B , RETURN.

LOMEM imm & def

```
lOMEM:aexpr
```

Set de laagst beschikbare geheugenplaats , die beschikbaar is voor BASIC. Normaal wordt LOMEM automatisch geïnitieerd vlak voor executie van een programma. LOMEM krijgt dan een waarde gelijk aan het einde van het programma.

Als LOMEM een grotere waarde krijgt dan HIMEM , verschijnt :

?OUT OF MEMORY ERROR

Ook als LOMEM lager wordt gemaakt , dan de hoogste geheugenlokatie , die door het systeem in gebruik is (2051) , wordt deze foutmelding gegeven.

LOMEM wordt gereset door NEW , DEL , of door verandering en toevoeging van regels aan een programma.

LOMEM wordt ook gereset door RESET , CTRL - B , return.

Als LOMEM hoger is , dan de hoogste geheugenplaats van het systeem en van lopende BASIC programma's , wordt de waarde niet gewijzigd door RUN.

De waarde LOMEM staat op de geheugenplaatsen 105 en 106.

Als LOMEM gewijzigd wordt kan dit alleen maar een waarde zijn , die groter is dan de oude waarde.

?OUT OF MEMORY ERROR

wordt gegeven als hieraan niet is voldaan.

Het wijzigen van LOMEM tijdens de program - executie kan ertoe leiden , dat het programma niet meer korrekt wordt voortgezet.

USR imm & def

usr (aexpr)

Met deze routine kan een parameter worden doorgegeven naar een machinetaal subroutine.

De waarde van aexpr wordt geplaatst in de floating accumulator (lokatie \$9D tot \$A3) en er wordt een JSR 000A uitgevoerd.

De gebruiker moet er dus voor zorgen , dat op die plaats een JMP naar een machinetaal programma staat (lokatie 000A tot 000C hex).

Een door de subroutine terug te geven waarde moet geplaatst worden in de floating accumulator.

Om een 2-byte integer te verkrijgen van / aexpr / in de floating accumulator , moet in de subroutine een JSR E10C uitgevoerd worden.

De integer waarde wordt geplaatst in de adressen A0 (meest significante byte) en A1 (minst significante byte)

Om een integer te converteren naar zijn floating point equivalent , zodat de subroutine de waarde aan het BASIC programma kan teruggeven , moet een JSR E2F2 uitgevoerd worden. De integer waarde moet daartoe staan in register A (meest significante byte) en Y (minst significante byte).

De terugkeer naar APPLESOFT vindt plaats door een RTS instructie.

2. Editing command's

LIST imm & def

```
list [rnum1] [-rnum2]  
list [rnum1] [,rnum2]
```

Als rnum1 noch rnum2 gegeven is , wordt het gehele programma gelist.

Als rnum1 aanwezig is , of rnum1 is gelijk aan rnum2 , dan wordt alleen het betreffende regelnummer afgedrukt.

Als rnum1 vooraf wordt gegaan door een - of , wordt het programma gelist van rnum1 tot het einde van het programma.

Als een , of - aanwezig is evenals rnum2 , worden de regels vanaf het begin van het programma tot rnum2 afgedrukt.

Zijn zowel rnum1 als rnum2 gegeven , dan wordt gelist vanaf rnum1 tot rnum2.

Als meer dan een regel moet worden gelist en het gespecificeerde rnum1 is niet in het programma aanwezig , dan wordt gelist vanaf de regel met het eerste regelnummer groter dan rnum1.

Als meer dan een regel gelist moet worden en rnum2 komt niet voor in het programma , dan wordt gelist tot en met de regel met het eerste regelnummer kleiner dan rnum2. De geliste regels zien er meestal anders uit , dan ze zijn ingevoerd.

Niet significante spaties worden verwijderd , andere spaties worden toegevoegd.

Het listen van een programma kan worden onderbroken met een CTRL-C , RESET of CTRL-S. In het laatste geval kan verder gelist worden door opnieuw CTRL-S te typen.

DEL imm & def

```
del rnum1,rnum2
```

Dit commando verwijdert de gegeven regelnummers uit het programma. Het commando kan slechts in deze vorm , dus met beide regelnummers gespecificeerd , worden ingevoerd. Bij ontbreken van een der regelnummers zal een ?SYNTAX ERROR worden gegeven.

Een enkele regel kan verwijderd worden door voor rnum1 en rnum2 dezelfde regel te specificeren. Eenvoudiger is het echter in dat geval het regelnummer in te typen gevolgd door RETURN.

Als DEL in een programma is opgenomen , wordt het commando uitgevoerd en de executie stopt. Het is dan niet meer mogelijk met CONT het programma voort te zetten.

REM imm & def

rem. karakter.

REM dient om het mogelijk te maken opmerkingen (REMarks)
in een programma op te nemen. Achter REM mogen alle
karakters , inclusief BASIC commando's worden opgenomen.
Een REM wordt afgesloten door return. REM statements
worden niet uitgevoerd.
De computer zal achter REM 1 spatie toevoegen , ongeacht
het aantal spaties dat daar al stond.

VTAB imm & def

vtab aexpr

Bij uitvoering van dit commando zal de cursor gaan naar
regel / aexpr /. De bovenste regel van het scherm is
regel 1 , de onderste regel 24. VTAB regelt de verticale
tabulatie.
Wordt een andere waarde gespecificeerd , dan tussen 1 en
24 volgt op het scherm :

?ILLEGAL QUANTITY ERROR

HTAB imm & def

htab aexpr

HTAB verplaatst de cursor naar positie / aexpr / van de
regel , waar de cursor momenteel staat. / aexpr / moet
liggen tussen 0 en 255 , anders wordt de foutmelding :

?ILLEGAL QUANTITY ERROR

gegeven.

Het is dus mogelijk de cursor meer dan 1 regel te
verplaatsen.

De regel waar de cursor staat heeft posities 1 t/m 40 ,
de volgende 41 t/m 80 enz.

HTAB 0 zet de cursor op positie 256.

TAB imm & def

tab (aexpr)

TAB kan alleen in PRINT statements worden gebruikt en aexpr moet tussen haakjes staan.
TAB verplaatst de cursor alleen dan , als de linker zijkant van het tekstvenster plus / aexpr / een positie geeft , die rechts van de huidige cursorpositie ligt.
TAB verplaatst de cursor nooit naar links (gebruik hiervoor HTAB).
Als TAB ertoe leidt , dat de cursor over de rechterzijkant van het tekstvenster gaat , wordt er aan de linkerkant van de volgende regel verdergegaan.
Voor de grenzen van / aexpr / geldt hetzelfde als voor HTAB.

POS imm & def

pos (expr)

Geeft de aanwezige cursorpositie in verhouding tot de linker marge van het tekstvenster.
Expr speelt geen rol in dit commando , maar moet wel een geldige waarde , naam van variabele , of expressie zijn.

De linkermarge heeft positie 0.
Merk op , dat bij HTAB en TAB de positie op het scherm genummerd zijn vanaf 1 terwijl POS nummert vanaf 0.

SPC imm & def

spc (aexpr)

SPC kan alleen in een PRINT statement gebruikt worden.
Geeft / aexpr / spaties vanaf het laatst afgedrukte karakter , of vanaf de linker marge van het tekstvenster.

SPC(0) geeft geen spaties.
/ aexpr / moet liggen tussen 0 en 255 , maar verschillende SPC commando's mogen na elkaar gebruikt worden om grotere spatiering te krijgen.
Als aexpr een 'real' is , wordt hij geconverteerd naar integer.

HOME imm & def

home

Geen parameters.

Dit commando verplaatst de cursor naar de bovenste , meest linkse positie van het tekstvenster. Alle tekst verdwijnt van het scherm.

Dit commando is identiek aan ESC , Shift - P , return , of CALL -936.

CLEAR imm & def

clear

Geen parameters.

Maakt alle variabelen , arrays en strings nul. Pointers en stacks worden gereset.

FRE imm & def

fre (expr)

FRE geeft een waarde terug gelijk aan het aantal bytes dat nog beschikbaar is in het geheugen. Als dit getal groter is dan 32767 , zal een negatief getal worden gegeven. Door hier 65536 bij op te tellen , verkrijgt men de werkelijke waarde. Als HIMEM op een waarde is gezet , die groter is dan de maximale geheugencapaciteit van de computer , kan FRE een niet zinvolle waarde teruggeven.

Als de inhoud van strings tijdens de programma - executie verandert , wordt de oude string niet verwijderd. Er wordt voor de nieuwe string een nieuwe file geopend. Bijvoorbeeld als A\$ de waarde "SALARIS" had en gelijk wordt gemaakt aan "GEBBOORTEDATUM" , wordt de string "SALARIS" niet uit het geheugen verwijderd. APPLESOFT zal automatisch 'opruiming houden' als de vrije ruimte opraakt.

Maar dit kan te laat komen , omdat machinetaal programma's , of graphic buffers dan al overschreven kunnen zijn.

Deze 'opruiming' kan eerder worden uitgevoerd door een FRE commando.

De expr tussen haakjes heeft geen betekenis , maar moet een geldige expressie of waarde zijn.

FLASH imm & def

flash

Dit commando is een zgn. video output commando.
Het heeft alleen effect bij karakters , die op de beeldbuis te zien zijn.
Werkt dus NIET tijdens het intypen van karakters en op karakters , die al op het scherm aanwezig waren.
FLASH heeft tot gevolg , dat de tekst knippert op het scherm : zwart op een witte en vervolgens wit op een zwarte achtergrond.

INVERSE imm & def

inverse

Dezelfde overwegingen als bij FLASH.
Output karakters worden echter zwart gedisplayed op een witte achtergrond.

NORMAL imm & def

normal

NORMAL zet de output mode weer normaal : wit op zwarte ondergrond.

SPEED imm & def

speed = aexpr

SPEED bepaalt de snelheid , waarmee karakters naar het beeldscherm , of naar een ander device worden gezonden.

De laagste snelheid is 0 , de hoogste 255.
Waarden buiten deze range veroorzaken een

?ILLEGAL QUANTITY ERROR

3. Over array's en strings

Tot zover heb je kennis gemaakt met de gewone variabele , om gegevens achter te verbergen (weet je nog : A = 10).

Een veel uitgebreidere , maar afwijkende mogelijkheid daartoe bood de string (A1\$ = C\$ + "MOOI WEER"). Er is nog een derde mogelijkheid voor het opslaan van gegevens en nu voor rijen getallen : de array (spreek uit : 'erreeh'). Dat kan van belang zijn voor een bedrijfsinventarisatie , het weergeven van sportuitslagen en statistieken.

De naam van de array is elke legale naam voor een variabele.

Een uitstekend voorbeeld is A. Om nu het juiste element uit een datalist (een lijst met getallen) te kiezen , geven we A een 'wijzer', of subscript. Het geweldige schuilt hem in het feit , dat het subscript enkel beperkt wordt door de geheugengrootte van je APPLE-II. Je kunt de array danook enorm dimensioneren , een geweldige grootte geven. Zo van DIM A(400). Hetgeen voor de APPLE-II inhoudt , dat er een array komt van A (0) tot en met A(400).

In het geheugen houdt hij daar heel handig ruimte voor vrij.

En na de A zijn er nog 25 letters in het alfabet. Probeer het volgende programma te doorgronden aan de hand van dit handboek :

```
10 DIM A(5)
20 FOR L = 1 TO 5
30 READ A(L)
40 NEXT L
50 DATA 10,20,30,40,50
60 FOR N = 1 to 5
70 PRINT N,A(N)
80 NEXT
```

Let op de FOR. . . NEXT lus , of 'loop' (spreek uit : loop) en de combinatie READ (= lees). . . . DATA (= gegevens).

Nu is het bovendien mogelijk complexe rijen getallen te verwerken , dus 2 of meer kolommen. Zo'n geheel van getallen noemen we een 'matrix' en ons oefenprogramma is daarvan eigenlijk al een eenvoudig voorbeeld.

De matrix is 2 horizontale en 5 verticale dimensies groot : hij is 2 op 5. Je kunt die ingewikkelde dimensies vooraf aan de APPLE-II opgeven , omdat APPLESOFT er anders van uitgaat , dat het maximale subscript (de 'index', volgens wiskundigen) 10 is. per dimensie. En nu wilde je net een matrix invoeren met 88 dimensies !! Eh , dat is overigens het maximum , bij 89 dimensies volgt een :

?OUT OF MEMORY ERROR

Een voorbeeld van zo'n brede dimensionering is :

100 BOEKEN (5,5,2)

Array elementen worden middels het RUN statement , of d. m. v. CLEAR op 0 gesteld. Dat is van belang voor het (later ?) doorzien van de READ. . . . DATA combinatie.

Samenvattend kun je zeggen , dat de array naast de string en de gewone variabele een geweldig hulpmiddel is voor de snelle verwerking van cijfermatige data ; dat hij vooraf gedimensioneerd moet worden in 1 of meer dimensies (matrix) , doch dat de geheugengrootte de grens van zijn mogelijkheden vormt.

DIM imm & def

dim var(subscr), { var(subscr) } .

Het DIM statement reserveert geheugenplaatsen voor arrays met de naam var.

Twee bytes worden gereserveerd voor de naam , twee voor de grootte in bytes , een byte voor de dimensie en twee voor elke dimensie.

Subscript lopen van 0 tot en met / subscr /.

Dus DIM DEMO(4,5,3) reserveert $5 \times 6 \times 4 = 120$ geheugenplaatsen.

Het maximale aantal dimensies van een array is 88. Een grotere dimensie geeft :

?OUT OF MEMORY ERROR

Als een array element gebruikt wordt voordat de DIMensionering van het array heeft plaatsgevonden , dan reserveert APPLESOFT 10 plaatsen per dimensie voor die array. Het gebruik van een arrayvariabele met een subscript groter dan / subscr / , of met meer dimensies , dan gedefinieerd , geeft dan

?BAD SUBSCRIPT ERROR

Wordt het DIM statement gebruikt voor een array , waarvoor al een DIM statement was gebruikt , of die door het systeem is gedefinieerd , dan verschijnt de melding :

?REDIM'D ARRAY ERROR

De afzonderlijke strings in een string array worden niet gedimensioneerd. Een string mag maximaal 255 karakters bevatten.

Array elementen worden op 0 gesteld door een RUN of een CLEAR commando.

LEN imm & def

len(sexpr)

Deze functie geeft een waarde , die gelijk is aan het aantal karakters van / sexpr /. Als dat een concatenatie (optelling) is van strings , waarvan het aantal karakters samen groter is dan 255 , dan volgt een

?TOO LONG ERROR

STR\$ imm & def

str\$(aexpr)

Deze functie converteert / aexpr / naar een string , die / aexpr / representeert.
Als / aexpr / groter is , dan de waarde toegestaan voor reals , wordt de foutmelding :

?OVERFLOW ERROR

gegeven.

VAL imm & def

val(sexpr)

Deze functie doet precies het omgekeerde van STR\$.
VAL tracht de / sexpr / te interpreteren als een real , of als een integer.
Het eerste karakter van de string (afgezien van spaties) moet een toegestaan karakter zijn , om een real of integer te vormen (0 t/m 9 + -. E spatie) , anders wordt een 0 teruggegeven.
Elk volgend karakter wordt op dezelfde manier beoordeeld, tot een karakter voorkomt , dat niet aan de gestelde eis voldoet.
Vanaf dat punt worden verdere karakters genegeerd.
Als sexpr een stringconcatenatie is met meer dan 255 karakters , dan wordt de foutmelding

?STRING TOO LONG ERROR

gegeven.

Als de absolute waarde van het verkregen getal groter is dan 10^{38} , of het getal heeft meer dan 38 karakters , inclusief voorafgaande nullen , dan wordt gedisplayed :

?OVERFLOW ERROR

CHR\$ imm & def

chr\$(aexpr)

Deze functie geeft een ASCII karakter , waarvan de waarde overeenkomt met / aexpr /.
/ aexpr / moet liggen tussen 0 en 255 , anders wordt de foutmelding

?ILLEGAL QUANTITY ERROR

geprint.

Reals worden geconverteerd naar integers.

ASC imm & def

asc(sexpr)

Omgekeerde functie van CHR\$.

Geeft de ASCII waarde van het eerste karakter van /sexpr/.

Op de APPLE-II zijn de ASCII codes van 96 tot 255 duplicaten van 0 tot 95.

Dus CHR\$(65) geeft een A , maar ook CHR\$(193) geeft een A.

In het ene geval zal de ASC functie 65 teruggeven , in het andere geval 193.

Als sexpr een stringconstante is , dan moet deze tussen " " - tekens staan en mogen er geen " " in de string voorkomen. Als de string nul is geeft dat de foutmelding:

?ILLEGAL QUANTITY ERROR

ASC("CTRL- ") geeft een

?SYNTAX ERROR

LEFT\$ imm & def

left\$(sexpr,aexpr)

Geeft de eerste / aexpr / karakters van / sexpr /.

Als aexpr een real is , wordt hij geconverteerd naar een integer.

Een

?ILLEGAL QUANTITY ERROR

wordt gegeven , als / aexpr / kleiner is dan 1 of groter dan 255.

Als / aexpr / groter is dan LEN (sexpr) , wordt de gehele string teruggegeven.
Als \$ in het commando wordt vergeten , beschouwt APPLESOFT LEFT als een rekenkundige variabele en wordt de foutmelding

?TYPE MISMATCH ERROR

gegeven.

RIGHT\$ imm & def

right\$(sexpr,aexpr)

De / aexpr / meest rechtse karakters van / sexpr / worden teruggegeven.
Verder gelden dezelfde overwegingen als bij LEFT\$.

MID\$ imm & def

mid\$(sexpr,aexpr1 f,aexpr2ij)

MID\$ met alleen aexpr1 geeft de stringexpressie vanaf karakter / aexpr1 / , tot het eind van de string.
MID\$ met zowel aexpr1 als aexpr2 geeft de string terug vanaf karakter / aexpr1 / met een lengte van / aexpr2 / karakters.
Als / aexpr1 / groter is dan LEN(sexpr) dan wordt de nul string teruggegeven.
Als / aexpr1 / + / aexpr2 / groter is dan de lengte van de / sexpr / , wordt de gehele string teruggegeven.
Voor de omvang van de expressie geldt hetzelfde als bij LEFT\$ en RIGHT\$.

STORE imm & def

store avar

Dit commando schrijft een array met de naam ' avar ' naar de cassetteband , zonder de naam.

RECALL imm & def

recall avar

Dit commando laadt de array ' avar ' met de waarden vanaf de cassetteband.

De dimensies van de array , die met een STORE is weggeschreven , moeten identiek zijn aan die , waarmee ze met een RECALL worden geladen.

In het algemeen zal een RECALL naar een array met minder elementen dan het aantal elementen op de cassette een :

?OUT OF MEMORY ERROR

geven.

Heeft het ' RECALL array ' meer elementen dan op band staan , dan zal

ERR

verschijnen , maar de programma-executie gaat gewoon door.

Alleen integer en real arrays mogen weggeschreven worden.

Stringarrays moeten met de ASC functie geconverteerd worden.

Hoewel bij STORE en RECALL alleen de naam van de array wordt gegeven en geen subscripts , zal alleen het array met die naam worden weggeschreven en niet een variabele met dezelfde naam.

Zowel bij STORE als RECALL klinkt aan het begin en einde van de opname een ' biep ' uit de luidspreker.

Als wordt geprobeerd een STORE of RECALL te doen met een array , die nog niet gedimensioneerd is , dan verschijnt de foutmelding :

?OUT OF MEMORY ERROR

STORE en RECALL kunnen alleen onderbroken worden door RESET.

4. Input - Output commando ' s

Een vooraf ingesteld programma is natuurlijk wel leuk. Vooral om de loop van het programma te bestuderen. Het verveelt echter snel en bovendien zijn veel in te voeren gegevens afhankelijk van verschillende omstandigheden.

Wat kopen we ervoor , wanneer we de APPLE-II zo programmeren , dat er elke keer een uitgebreide tabel met gegevens ontstaat , wanneer we maar behoefte hebben aan 1 specifieke waarde ?

Kortom , je wilt met de APPLE-II kunnen communiceren.

Tja , daar is hij volgens ons voor ; hij is zeker geen oliedom rekenmachientje ! Een beslissend statement in het communicatieproces is INPUT.

Op het moment , dat je APPLE-II dit statement tegenkomt , stopt hij de programma - executie en wacht hij , tot je het gewenste gegeven hebt ingevoerd.

Input wil zeggen : invoer , of ' voer wat in '.

Hier hebben we een aardig oefenprogramma , dat willekeurige scores bij een sport tot een totaal uitslag verwerkt.

Probeer aan de hand van dit handboek de werking van het programma te analyseren :

```
10 PRINT "SPORT UITSLAGEN/WEDSTRIJDSCORE"
20 PRINT
30 PRINT "(AAN EINDE PROGRAMMA TYPE 0)"
40 PRINT
45 PRINT "AANTAL DEELNEMERS";
50 INPUT N
60 IF N=0 THEN 270 : REM 270 IS END
70 DIM A$(50) : REM BEPERKING ARRAY DEELNEMERS
80 FOR I= 1 TO N
90 PRINT"DEELNEMER ";I;
100 INPUT A$(I)
110 NEXT I
120 FOR I= 1 TO N
130 FOR J= 1 TO N-1 : REM ALFABETISEERLUS
140 A$=A$(J)
150 B$=A$(J+1)
160 IF A$ > B$ THEN 190
170 A$(J)=B$
180 A$(J+1)= A$
190 NEXT J
200 NEXT I
210 PRINT
220 FOR I=1 TO N : REM OUTPUTLOOP
230 PRINT A$(I)
240 NEXT I
250 PRINT
260 GOTO 40 : REM TERUGKEER NR BEGIN
270 END
```

Hoe gebruik je dit programma ?
 Voer in , wanneer de APPLE-II het INPUT - vraagteken met
 cursor laat zien :

0 JANSEN
 123 GOOR
 012 CORNELISSE

De uitslagen / scores worden keurig van hoog tot laag
 afgedrukt.
 Wil je met dit programma namen alfabetiseren, verander
 dan regel 160 :

160 IF A\$ < B\$ THEN 190
 Veel genoeg met dit programma !

programma te analyseren :
 probeer aan de hand van dit handboek de werking van het
 verwerkt.
 wilkeurige scores bij een sport tot een totaal uitslag
 Hier hebben we een aardig oefenprogramma , dat
 input wil zeggen : invoer , of ' voer wat in ' .
 het gewenste gegeven hebt ingevoerd .
 stopt hij de programma - executie en wacht hij , tot je
 op het moment , dat je APPLE-II dit statement tegenkomt ,
 communicatieproces is INPUT . Een beslissend statement in het
 oledom rekenmachientje ! Een beslissend statement in het
 ja , daar is hij volgens ons voor : hij is zeker geen
 Kortom : je wilt met de APPLE-II kunnen communiceren

```

270 END
260 GOTO 40 : REM TERUGKEER NA BEGIN
250 PRINT
240 NEXT I
230 PRINT A$(I)
220 FOR I=1 TO N : REM OUTPUTLOOP
210 PRINT
200 NEXT I
190 NEXT J
180 A$(J+1)=A$
170 A$(J)=B$
160 IF A$ > B$ THEN 190
150 B$=A$(J+1)
140 A$=A$(J)
130 FOR J=1 TO N-1 : REM ALFABETISERING
120 FOR I=1 TO N
110 NEXT I
100 INPUT A$(I)
90 PRINT"DEELNEMER " ; I
80 FOR I=1 TO N
70 DIM A$(50) : REM BEPERKING ARRAY DEELNEMERS
60 IF N=0 THEN 270 : REM 270 IS END
50 INPUT N
45 PRINT "AANTAL DEELNEMERS"
40 PRINT
30 PRINT "(AAN EINDE PROGRAMMA TYPE 0)"
20 PRINT
10 PRINT "SPORT UITSLAGEN\WEDSTRIJDSCORE"
```

INPUT def

```
input [string:] var [,var]
```

Als de string wordt weggelaten , wordt een vraagteken op het scherm gedisplayed en gewacht tot je een getal , of een string hebt ingevoerd.

Als de string wel is gespecificeerd , wordt hij zonder extra spatiering exact als aangegeven afgedrukt. Het vraagteken wordt dan niet afgedrukt.

Er mag slechts een string gebruikt worden en deze MOET afgesloten zijn met :.

INPUT accepteert alleen reals , integers en strings , geen expressies.

Als hieraan niet voldaan wordt , zal meestal de foutmelding

REENTER

worden gegeven , waarna de waarde opnieuw kan worden ingevoerd.

Als var een stringvariabele is , dan kan een literal , of een string worden ingevoerd. Als een literal wordt ingevoerd , mag het eerste niet - spatie karakter geen " zijn. Is dat wel het geval , dan worden de ingevoerde karakters als string beschouwd. Er mogen dan geen " in de string voorkomen , maar de string moet wel afgesloten worden met een ". Als een literal wordt ingevoerd mogen binnen de literal wel " - tekens voorkomen.

CTRL - X en CTRL - M binnen een ingevoerde string zijn niet toegestaan.

Als het systeem een arithmetische waarde verwacht en RETURN wordt direkt ingedrukt dan krijg je

?REENTER

Als een string wordt verwacht en de RETURN toets wordt meteen bediend , dan wordt de invoer als nul string beschouwd.

Als verschillende variabelen in het INPUT statement worden gespecificeerd , moet de bijbehorende waarden in de juiste volgorde worden ingevoerd.

Ze kunnen worden gescheiden door komma ' s , of door RETURNS ' s. In het laatste geval zal , zolang niet alle invoer is gegeven ,

??

op het scherm verschijnen.

Stringvariabelen en arithmetische variabelen mogen gemengd in een INPUT statement voorkomen , maar de juiste waarden moeten in de juiste volgorde worden ingevoerd.

Als een : wordt ingevoerd in een INPUT response die niet met ? begon , dan worden alle daarna ingetikte karakters genegeerd , ook de komma ' s. Er kan dan alleen een nieuwe response cyclus worden geforceerd door een RETURN.

Worden meer waarden ingevoerd , dan er variabelen waren gespecificeerd in het INPUT statement , volgt de melding:

[rv.] rv [string] input
?EXTRA IGNORED

maar de programma executie zal gewoon doorgaan.
Als een ; of een , is ingetikt als eerste karakter van een INPUT - response , dan wordt dit beschouwd als 0 of nul string.
CTRL - C kan een input onderbreken , maar alleen als dit bij de eerste responsecyclus gebeurt . Na de CTRL - C moet een RETURN gegeven worden .
Een poging het programma daarna met een CONT voort te zetten geeft een

?SYNTAX ERROR

INPUT kan , zoals hiervoor is aangegeven , niet in de immediate mode worden gebruikt . Een poging hiertoe geeft:

?ILLEGAL DIRECT ERROR

GET def

get var

Deze functie accepteert een karakter van het keyboard (= toetsenbord) , zonder het weer te geven en zonder dat de return toets bediend hoeft te worden .
CTRL , Shift - P geeft een nul karakter .
CTRL - H of de <- toets geeft ook een nul karakter .
CTRL - C wordt behandeld als elk ander karakter .
programma executie wordt niet afgebroken .

Als var een rekenkundige variabele is , mogen in het algemeen alleen de cijfers 0 t/m 9 worden gebruikt , omdat andere karakters foutmeldingen kunnen veroorzaken , of ze geven de waarde 0 terug .
We raden aan om var als stringvariabele te gebruiken .

op het scherm verschijnen .
Stringvariabelen en arithmetische variabelen worden gemeenschappelijk in een INPUT statement voorkomen , maar de juiste volgorde worden ingevoerd .
Als een ; wordt ingevoerd in een INPUT response die niet met ; begon , dan worden alle daarna ingetikte karakters genegeerd , ook de komma ' , . Er kan dan alleen een nieuwe responsecyclus worden geforceerd door een RETURN .

DATA def

```
data [lit: string: real: integer] [ { , [lit:
string: real: integer]} ] .
```

Dit statement definieert een lijst van elementen , die gebruikt kunnen worden met het READ statement. In volgorde van regelnummer voegen DATA statements elementen toe aan de lijst , die door eerdere DATA statements is gevormd.

DATA statements hoeven niet perse voor READ statements te worden geplaatst , maar mogen overal in het programma staan. Voor DATA elementen , die gelezen worden met een READ , gelden (meestal) dezelfde regels als voor data , die worden binnengehaald met het INPUT statement.

Een : mag echter nooit voorkomen in een DATA element.

Als CTRL - C een DATA element is , wordt programma executie niet gestopt bij een READ van dit element.

Met alleen deze uitzondering , gelden voor DATA stringelementen dezelfde regels als voor de INPUT van strings.

Als de lijst van elementen in een DATA statement ' niet bestaande ' elementen bevat , wordt een 0 , of de nulstring verondersteld.

Dat gebeurt in de volgende gevallen :

- 1) Er komt geen niet - spatie karakter voor tussen DATA en RETURN.
- 2) De komma is het eerste niet - spatie karakter na DATA
- 3) Er staat geen niet - spatie karakter tussen twee komma ' s
- 4) Een komma is het laatste niet - spatie karakter voor RETURN.

READ imm & def

```
read var [ { , var } ]
```

Als het eerste READ statement in een programma wordt uitgevoerd , zal de eerste variabele de eerste waarde krijgen uit de tabel van DATA elementen , de tweede variabele de tweede waarde enz.

Als de executie van het READ statement is afgelopen , blijft er EEN POINTER staan achter het laatst gelezen element van de lijst. Bij een volgende READ wordt dan DE VOLGENDE waarde uit de lijst gelezen.

RUN en RESTORE resetten de DATA pointer naar het eerste element van de DATA - tabel.

Als geprobeerd wordt meer elementen te lezen , dan er in de tabel zijn , wordt de foutmelding :

?OUT OF DATA ERROR IN XXXX

afgedrukt.

XXXX geeft het regelnummer aan , waar het READ statement staat , dat de fout veroorzaakte. In de immediate mode kunnen er alleen DATA elementen worden gelezen , als er in een voorafgaand programma elementen in een DATA lijst zijn opgenomen. Deze elementen kunnen gelezen worden , ook al heb je dat programma nooit geRUNd. Als een programma in immediate mode wordt uitgevoerd , wordt de DATA pointer NIET gereset op het eerste element.

RESTORE imm & def

restore

Dit commando zet alleen de DATA pointer terug naar het eerste element van de DATA lijst.

PRINT imm & def

```
print [ { { expr } [ { , : ; [ { expr } ] } ] [ , ; ]
print { } } .
print { , } .
```

In plaats van PRINT mag het vraagteken gebruikt worden. Het wordt echter wel geLIST als PRINT.

Met PRINT wordt overgegaan naar een nieuwe regel.

Als expressies worden gespecificeerd , dan worden de waarden hiervan afgedrukt.

Als er geen ; of , aan het einde van een PRINT statement staat , wordt overgegaan naar een nieuwe regel , nadat het laatste item is gePRINT.

Als een item wordt gevolgd door een komma , wordt het volgende te printen item afgedrukt op de eerste positie van het volgende TAB veld.

Als een item in de lijst wordt gesloten met ; wordt het volgende item uit de lijst afgedrukt achter het vorige , maar zonder tussenliggende spatie.

Er mogen achter een PRINT karakters en statements voorkomen tot maximaal 239 tekens.

IN# imm & def

in# aexpr

Selecteert output van I / O slot / aexpr / .
Dit statement wordt gebruikt , om aan te geven welk ' periferale device ' (bijv. een disc - drive) gebruikt wordt , om input te leveren bij executie van een INPUT statement.

Gespecificeerd mogen worden slot 0 t / m 7.

IN#0 betekent dat input van het APPLE keyboard komt.

Als in een gespecificeerd slot geen device aanwezig is , ' hangt ' het systeem. Alleen een RESET (CTRL - C) herstelt deze toestand.

Als / aexpr / groter is dan 255 , of kleiner dan 0 verschijnt de foutmelding :

?ILLEGAL QUANTITY ERROR.

Als / aexpr / ligt tussen 8 en 255 , dan kunnen er onvoorspelbare dingen gebeuren.

PR# imm & def

pr# aexpr

PR# selecteert een slot voor output van PRINT statements.

/ aexpr / moet liggen tussen 0 en 7.

PR#0 selecteert de video - uitgang.

Verder dezelfde overwegingen als bij IN#.

LET imm & def

```
[let] avar [subscr] = aexpr  
[let] avar [subscr] = sexpr
```

LET wordt gebruikt voor toewijzing van een waarde aan een variabele.

De variabele mag een arithmetische (= rekenkundige) , of een stringvariabele zijn.

De bijbehorende expressie moet dezelfde soort zijn , anders verschijnt :

?IYPE MISMATCH ERROR

op het scherm.

LET mag vrijblijvend worden gebruikt.

DEF def

```
def FN naam ( real avar ) = aexpr1
```

Maakt het mogelijk een functie te definiëren in BASIC programma 's.

Als de functie eenmaal is gedefinieerd , mag deze zo vaak als gewenst worden aangeroepen met FN.

Voor de naam geldt dezelfde overweging als bij variabelen, d. w. z. de eerste twee karakters dienen bepalend te zijn.

Functies kunnen opnieuw worden gedefinieerd.

Als real avar wordt bij definitie een voorlopige waarde (= dummy) meegegeven. Als de functie later aangeroepen wordt , gebeurt dat met een actuele waarde.

FN imm & def

```
fn naam ( aexpr2 )
```

Met FN wordt een functie aangeroepen , die met DEF is gedefinieerd.

De waarde van aexpr2 wordt op alle plaatsen , waar in aexpr1 (zie DEF) real avar voorkomt , ingevuld. Daarna wordt aexpr1 geevalueerd.

Voorbeeld :

```
100 DEF FN DEMO(W)=2*W+W
110 PRINT FN DEMO(23)
RUN
69
```

Dit resultaat kwam tot stand , doordat voor de formele parameter in regel 110 de waarde 23 werd ingevuld , zodat de waarde van de functie DEMO wordt $2 \cdot 23 + 23 = 69$.

Als een functie wordt aangeroepen , voordat hij met een DEF statement is geDEFinieerd , dan verschijnt de melding:

?UNDEF'D FUNCTION ERROR

op het scherm.

String en integer functies zijn niet toegestaan.

5. Flow control commando ' s

GOTO imm & def

goto rnum

Veroorzaakt een sprong naar het gespecificeerde regelnummer.

Als deze regel niet bestaat veroorzaakt dat een :

?UNDEF'D STATEMENT ERROR IN XXXX

waarbij XXXX het regelnummer is van het GOTO statement , die de foutmelding veroorzaakt

IF imm & def

```
if expr then instructie [ { : instructie } ]  
if expr then [ goto ] rnum  
if expr [ then ] goto rnum
```

Als / expr / ongelijk is aan 0 , wordt de waarde als TRUE (= waar) beschouwd en wordt de instructie (of worden de instructies) achter THEN uitgevoerd.

Als de waarde van expr 0 is , dan worden de instructies achter THEN genegeerd en wordt de eerstvolgende regel uitgevoerd.

Als bij toepassing van dit statement in immediate mode de waarde van expr 0 is , worden alle volgende statements genegeerd.

Een THEN zonder IF , of een IF zonder THEN zal een

?SYNTAX ERROR

veroorzaken.

In IF statements mag expr geen stringexpressie zijn , hoewel daar geen foutmelding op wordt gegeven.

Komt ' IF THEN ' meer dan twee keer voor in een programmaregel , dan volgt de foutmelding :

?FORMULA TOO COMPLEX ERROR.

De letter A voor THEN veroorzaakt problemen.

IF BETA THEN 230 wordt door APPLESOFT geïnterpreteerd als IF BET AT HEN230 en geeft een ?SYNTAX ERROR.

De volgende statements zijn identiek :

```
IF A=3 THEN 160  
IF A=3 GOTO 160  
IF A=3 THEN GOTO 160
```

FOR imm & def

```
for real avar = aexpr1 to aexpr2. step aexpr3.
```

Real avar wordt gelijk gemaakt aan aexpr1 en de statements , die de FOR statements volgen , worden uitgevoerd tot de statement NEXT real avar. Hierbij dient de naam van real avar in de FOR en NEXT statement dezelfde te zijn.

Vervolgens wordt real avar verhoogd met / aexpr3 / (en bij afwezigheid van ' STEP aexpr3 ' met 1) , pas daarna vergeleken met / aexpr2 /.

Als real avar groter is dan / aexpr2 / , wordt verder gegaan met het statement achter NEXT .

Als dit niet het geval is , dan wordt er terug gesprongen naar het statement volgend op het FOR. . . . statement.

Real avar MOET een real zijn , anders wordt de foutmelding

?SYNTAX ERROR

gegeven.

Daar real avar aan het einde van de lus pas verhoogd wordt en vergeleken met / aexpr2 / wordt de lus altijd tenminste een keer doorlopen.

FOR. . . . NEXT lussen mogen elkaar niet kruisen. Indien dit het geval is wordt de foutmelding

?NEXT WITHOUT FOR ERROR

gegenereerd.

Als meer dan 10 FOR. . . . NEXT lussen in elkaar genesteld worden , volgt de foutmelding

?OUT OF MEMORY ERROR.

Als het FOR statement in ' immediate mode ' wordt gebruikt , moeten het FOR statement , evenals NEXT in dezelfde regel voorkomen.

NEXT imm & def

```
next [ avar ]  
next avar [ { , avar } ]
```

NEXT vormt het einde van een FOR. . . . NEXT lus. Als avar niet is gespecificeerd , dan wordt ze beschouwd te behoren bij de laatst uitgevoerde FOR. . . . statement.

Verschillende FOR. . . . lussen kunnen met een enkele NEXT worden afgesloten. Wordt avar niet gespecificeerd, dan kan een conditionele sprong uit de lus later problemen geven, als avar niet op de juiste waarde wordt resteld.

GOSUB imm & def

gosub rnum

Veroorzaakt een sprong naar een subroutine , die begint op regel / rnum /.

Als een hierbij noodzakelijk RETURN statement wordt uitgevoerd , wordt het programma voortgezet bij het statement , dat op GOSUB volgt.

Elke keer wanneer GOSUB wordt uitgevoerd , wordt het terugkeerpunt in een afzonderlijke stack geplaatst , zodat de terugweg naar het programma weer kan worden gevonden.

Elke keer als een RETURN , of een POP wordt uitgevoerd , wordt het laatste adres van de stack verwijderd. Het POP statement moet telkens worden gebruikt bij een conditionele onderbreking van de GOSUB.

Als rnum niet in het programma voorkomt , verschijnt de melding

?UNDEF ' D STATEMENT ERROR IN XXXX

Op de plaats XXXX staat het regelnummer , waar het GOSUB statement voorkomt , die de foutmelding veroorzaakt. Als er meer dan 25 GOSUB ' s in elkaar genesteld zijn , dan leidt dat tot de

?OUT OF MEMORY ERROR.

RETURN imm & def

return

Dit is een sprong naar het statement , dat volgt op het laatst uitgevoerde GOSUB statement. Als er meer RETURN ' s worden opgedragen , dan er GOSUB ' s zijn , resulteert dit in

?RETURN WITHOUT GOSUB ERROR.

Na de uitvoering van RETURN is het bijbehorende terugkeeradres niet langer aanwezig in de stack.

POP imm & def

pop

POP heeft hetzelfde effect als RETURN met het verschil , dat de terugsprong niet wordt uitgevoerd. Het gevolg is , dat een volgende RETURN de programma - executie laat voortzetten na het voorlaatste uitgevoerde GOSUB. Het terugkeeradres van de laatste GOSUB is immers uit de stack verwijderd.

Als er meer POP ' s worden uitgevoerd dan er GOSUB ' s zijn , volgt een

?RETURN WITHOUT GOSUB ERROR.

ON def (GOTO / GOSUB)

on aexpr goto rnum ,rnum
on aexpr gosub rnum , rnum

Programma - executie wordt voortgezet op het gespecificeerde regelnummer met rangnummer / aexpr / in de lijst van regelnummers.

Als / aexpr / 0 is of groter dan het aantal gespecificeerde rnum ' s , wordt de executie voortgezet met het statement volgend op het ON. . . . statement. / aexpr / moet liggen tussen 0 en 255 , om de foutmelding

?ILLEGAL QUANTITY ERROR

te vermijden.

ONERR GOTO def

onerr goto rnum

Als er een fout optreedt , kan dit statement voorkomen , dat een foutmelding wordt afgedrukt en de programma - executie wordt onderbroken.

Als het commando wordt uitgevoerd , wordt een flag geset, die er bij een fout voor zorgt , dat een sprong wordt uitgevoerd naar rnum.

Met behulp van POKE 216,0 moet de flag gereset worden. Het ONERR statement moet worden geprogrammeerd , voordat een fout kan optreden , anders wordt de foutmelding terdege gegeven en het programma onderbroken.

Als een fout is opgetreden , kan je met PEEK(222) een ' foutnummer ' opvragen.

De code heeft de volgende betekenis :

0	NEXT WITHOUT FOR
16	SYNTAX
22	RETURN WITHOUT GOSUB
42	OUT OF DATA
53	ILLEGAL QUANTITY
69	OVERFLOW
77	OUT OF MEMORY
90	UNDEFINED STATEMENT
107	BAD SUBSCRIPT
120	REDIMENSIONED ARRAY
133	DIVISION BY ZERO
163	TYPE MISMATCH
176	STRING TOO LONG
191	FORMULA TOO COMPLEX
224	UNDEFINED FUNCTION
254	BAD RESPONSE TO INPUT STATEMENT
255	CTRL - C INTERRUPT ATTEMPTED

Je moet voorzichtig zijn met errorbehandeling na fouten , die optreden in FOR. . . NEXT lussen , en fouten in subroutines.

Het statement moet de lus hervatten door terug te springen naar het FOR , of naar het GOSUB statement. Als een programma na een fout wordt voortgezet bij NEXT , of RETURN , geeft dat de melding

```
?NEXT WITHOUT FOR ERROR of
?RETURN WITHOUT FOR ERROR
```

RESUME def

resume

Als dit commando wordt gebruikt aan het einde van een errorbehandelingsroutine , wordt het programma voortgezet, waar het onderbroken is ten gevolge van een fout en wel aan het begin van het statement.

Als RESUME wordt uitgevoerd , voordat een fout is opgetreden , resulteert dat in een

```
?SYNTAX ERROR IN 65278
```

Meestal zal het programma afgebroken worden.

Als een fout optreedt in een errorbehandelingsroutine waarin RESUME voorkomt , dan zal dit een oneindige lus veroorzaken.

RESET , CTRL - C , RETURN brengt je terug bij het programma.

6. Spelletjes , of Game control

PDL imm & def

pd1 (aexpr)

Deze funktie geeft een waarde terug tussen 0 en 255.
Deze waarde hangt af van de stand van een variabele
weerstand (potentiometer) , die aangesloten wordt op de
game I / O connector (zie deel ' HARDWARE ').

/ aexpr / geeft aan welke van de vier mogelijke ingangen
gelezen moet worden.

Het gebruik van andere waarden , dan 0 t / m 3 kan tot
gevolg hebben , dat bestaande programma ' s worden
gestoord.

De analoge spanning , die door de potmeter wordt
aangeboden , wordt door een analoog / digitaal omzetter
vertaald in een binaire waarde tussen 0 en 255. Deze
omzetter heeft na een conversie enige tijd nodig zich te
herstellen.

Daarom wordt aangeraden tussen twee PDL commando ' s
enige programmeerregels (bijvoorbeeld een delayloop als
T = 1 TO 25 : NEXT) op te nemen.

De paddles zijn als accessoires te verkrijgen.

HOOFDSTUK 6 INGEBOUWDE FUNKTIES

1. Wiskundige functies

APPLESOFT bevat een aantal ingebouwde mathematische functies.

Alle functies mogen daar worden gebruikt , waar expressies van hetzelfde soort ook mogen voorkomen.

SIN imm & def

sin (aexpr)

Deze functie berekent de sinus van / aexpr / radialen.

COS imm & def

cos (aexpr)

Deze functie berekent de cosinus van / aexpr / radialen.

TAN imm & def

tan (aexpr)

Deze functie berekent de tangens van / aexpr / radialen.

ATN imm & def

atn (aexpr)

Deze functie berekent de arctangens van / aexpr / in radialen.

De teruggegeven waarde ligt tussen $-\pi/2$ en $\pi/2$ radialen.

INT imm & def

int (aexpr)

Deze functie berekent de integer , die kleiner dan , of gelijk is aan / aexpr /.

RND imm & def

rnd (aexpr)

Deze functie geeft een random getal , dat groter of gelijk 0 is , maar kleiner dan 1.

Als / aexpr / groter is dan 0 , wordt elke keer als RND wordt gebruikt een nieuw getal gegenereerd.

Als / aexpr / kleiner is dan 0 wordt een pseudo random reeks geïnitieerd.

Met verschillende negatieve getallen zal een andere reeks worden verkregen.

De achtergrond hiervan is , dat men vaak wenst , dat programma' s met random getallen zich a.h.w.

'reproduceren' , d.w.z. dat elke keer als dat programma wordt gerUND dezelfde gemiddelde resultaten ontstaan. Vooral bij het zoeken naar fouten kan dit belangrijk zijn.

Als / aexpr / 0 is , wordt het laatst gegenereerde random getal opnieuw gegeven.

SGN imm & def

sgn (aexpr)

SGN geeft -1 als / aexpr / negatief is , 0 als / aexpr / 0 is en +1 als / aexpr / positief is.

ABS imm & def

abs (aexpr)

Geeft de absolute waarde van / aexpr /.

SQR imm & def

sqr (aexpr)

Geeft de positieve wortel van / aexpr /.
Deze functie werkt vaak sneller dan / aexpr / $^{.5}$

EXP imm & def

exp (aexpr)

Verheft $e(2.718289)$ tot de macht / aexpr /.

LOG imm & def

log (aexpr)

Geeft de natuurlijke logaritme (grondtal e) van / aexpr /.

De volgende , niet ingebouwde functies , kunnen met
bestaande functies eenvoudig worden gedefinieerd , als
DEF FN wordt gebruikt.

SECANS :

$\sec(x) = 1/\cos(x)$

COSECANS :

$\csc(x) = 1/\sin(x)$

COTANGENS :

$\cot(x) = 1/\tan(x)$

ARCSINUS :

$\text{asin}(x) = \text{atn}(x/\text{sqr}(-x*x+1))$

ARCCOSINUS :

$$\text{acos}(x) = -\text{atn}(x/\text{sqr}(-x*x+1)) + 1.5708$$

ARCCUSECANS :

$$\text{acs}(x) = \text{atn}(1/\text{sqr}(x*x-1)) + (\text{sgn}(x)-1)*1.5708$$

ARCSECANS :

$$\text{asc}(x) = \text{atn}(\text{sqr}(x*x-1)) + (\text{sgn}(x)-1)*1.5708$$

ARCCOTANGENS :

$$\text{atg}(x) = -\text{atn}(x) + 1.5708$$

SINUS HYPERBOLICUS :

$$\text{shb}(x) = (\exp(x) - \exp(-x))/2$$

COSINUS HYPERBOLICUS :

$$\text{chb}(x) = (\exp(x) + \exp(-x))/2$$

TANGENS HYPERBOLICUS :

$$\text{thb}(x) = -\exp(-x)/(\exp(x) + \exp(-x))*2+1$$

SECANS HYPERBOLICUS :

$$\text{sech}(x) = 2/(\exp(x) + \exp(-x))$$

COSECANS HYPERBOLICUS :

$$\text{csch}(x) = 2/(\exp(x) - \exp(-x))$$

COTANGENS HYPERBOLICUS :

$$\coth(x) = \exp(-x) / (\exp(x)) * 2 + 1$$

ARCSINUS HYPERBOLICUS :

$$\operatorname{asinh}(x) = \log(x + \operatorname{sqr}(x * x + 1))$$

ARCCOSINUS HYPERBOLICUS :

$$\operatorname{acosh}(x) = \log(x + \operatorname{sqr}(x * x - 1))$$

ARCTANGENS HYPERBOLICUS :

$$\operatorname{atanh}(x) = \log((1+x)/(1-x))/2$$

ARCSECANS HYPERBOLICUS :

$$\operatorname{asech}(x) = \log((\operatorname{sqr}(-x * x + 1) + 1) / x)$$

ARCCOSECANS HYPERBOLICUS :

$$\operatorname{acsech}(x) = \log(\operatorname{sgn}(x) * \operatorname{sqr}(x * x + 1) + 1 / x)$$

ARCCOTANGENS HYPERBOLICUS :

$$\operatorname{actanh}(x) = \log((x+1)/(x-1))/2$$

A MOD B :

$$\operatorname{mod}(a) = \operatorname{int}((a/b - \operatorname{int}(a/b)) * b + .05) * \operatorname{sgn}(a/b)$$

2. PEEKS , POKES en CALL' S

HEXADECIMAAL	COMMANDO	BESCHRIJVING
--------------	----------	--------------

DISPLAY controls

C050	POKE-16304,0	Set grafics mode
C051	POKE-16303,0	Set tekst mode
C052	POKE-16302,0	Reset gemengde tekst-grafics
C053	POKE-16301,0	Set gemengde tekst-grafics
C054	POKE-16300,0	Display page 1
C055	POKE-16299,0	Display page 2
C056	POKE-16298,0	Reset HGR
C057	POKE-16297,0	Set HGR

TEKST controls

0020	POKE 32,L1 L1 (0 - 39)	Zet linker marge op
0021	POKE 33,W1 W1 (L + W1 μ 40)	Zet venster breedte op
0022	POKE 34,T1 T1 (0 - 23)	Zet bovenkant venster op
0023	POKE 35,B1 B1 (0 - 23 , B1 \circ T1	Zet onderkant venster op
0024	CH=PEEK(36) positie POKE 36,CH CH (0 - 39)	Lees horizontale cursor Zet cursorpositie op
0025	CV=PEEK(37) POKE 37,CV CV (0 - 23)	Lees verticale cursor positie Zet cursorpositie op
0032	POKE 50,127 ook CALL -384 POKE 50,255 ook CALL -380	Set inverse flag ; Set normal flag ;
FC58	CALL -936	Esc , Shift P , Cursor home , wis scherm
FC42	CALL -958	ESC-F , wis scherm vanaf cursor tot einde van het scherm
FC9C cursor	CALL -868	ESC-E , wis scherm vanaf tot einde van de regel
FC66	CALL -922	CTRL-J , nieuwe regel
FC70	CALL -912	Scroll tekst 1 regel omhoog

Diversen

C030	X=PEEK(-16336)	Enkele puls naar speaker
	POKE -16336,0	idem
C000	X=PEEK(-16384)	Lees keyboard. Als X ° 127 dan
		is een toets ingedrukt
C010	POKE -16368,0	Reset keyboard strobe. Altijd
		noodzakelijk na een Lees
		keyboard
C061	X=PEEK(-16287)	SW 0 van de Game I / 0
		connector Als X ° 127 dan
		switch is aan
C062	X=PEEK(-16286)	SW 1 van de Game I / 0
		connector
C063	X=PEEK(-16285)	SW 2 van de Game I / 0
		connector
C058	POKE-16296,0	Reset Game I / 0 AN 0 output
C059	POKE-16295,0	Set Game I / 0 AN 0 output
C05A	POKE-16294,0	Reset Game I / 0 AN 1 output
C05B	POKE-16293,0	Set Game I / 0 AN 1 output
C05C	POKE-16292,0	Reset Game I / 0 AN 2 output
C05D	POKE-16291,0	Set Game I / 0 AN 2 output
C05E	POKE-16290,0	Reset Game I / 0 AN 3 output
C05F	POKE-16289,0	Set Game I / 0 AN 3 output
C020	PEEK(-16352)	Geeft puls op cassette recorder- uitgang
00D8	PEEK(216)	Lees ERR-flag als ° 127 dan
	gezet	
	POKE 216,0	Reset ERR-flag
00DE	X=PEEK(222)	X geeft errorcode

3. Automatische foutmeldingen

Nadat een fout is opgetreden , geeft APPLESOFT een foutmelding door en daarna geeft het je het toetsenbord terug. Waarden van variabelen , alsmede de programmatekst blijven onveranderd , maar het programma kan niet worden geCONTinueerd , terwijl alle GOSUB en FOR counters op 0 worden gezet.

Om te voorkomen , dat een lopend programma door een fout wordt afgebroken , kan het UNERR statement worden gebruikt , samen met een ' error handling routine'. Als een fout optreedt in de IMMEDIATE mode , wordt geen regelnummer gegeven.

IMMEDIATE executie	?XXX ERROR
DEFERRED executie	?XXX ERROR IN YYYY

In beide voorbeelden is XXX de specifieke foutmelding en YYYY het regelnummer , waar de fout is opgetreden.

In ' deferred mode' worden fouten vastgesteld , wanneer het programma wordt uitgevoerd.
Niet bij invoeren van programmatekst.

De foutmeldingen , die op kunnen treden zijn :

CAN ' T CONTINUE

Treedt op na een poging een programma te continueren , dat niet bestaat , of nadat een fout is opgetreden , of nadat in een afgebroken programma een of meer regels zijn gewijzigd of verwijderd.

DIVISION BY ZERO

Delen door 0 is niet toegestaan.

ILLEGAL DIRECT

INPUT , DEF , GET en DATA mogen niet als ' immediate' executie commando worden gebruikt.

ILLEGAL QUANTITY

De parameter meegegeven aan een mathematische , of string funktie was buiten de toegestane grootte.

- a) een negatieve array subscript
- b) LOG met een negatief argument
- c) SQR met een negatief argument
- d) A^B met A negatief en B geen Integer
- e) gebruik van MID\$, LEFT\$, RIGHT\$, WAIT , PEEK , POKE , TAB , SPC , ON.... GOTO , of een van de graphics funkties met een verkeerd argument.

NEXT WITHOUT FOR

De variabele in een NEXT statement correspondeert niet met een variabele in het FOR statement , die het laatst werd uitgevoerd.

Een naamloze NEXT correspondeert niet met een FOR statement.

OUT OF DATA

Er wordt een READ statement uitgevoerd , terwijl er geen elementen meer in de DATA lijst aanwezig zijn.

OUT OF MEMORY

Programma te groot.

Te veel variabelen.

FUR loops meer dan 10 diep genesteld.

GOSUB' s meer dan 24 diep genesteld.

Te gecompliceerde expressies.

Haakjes meer dan 36 diep genesteld.

Poging LOMEM te hoog te zetten.

Poging LOMEM lager te maken , dan oude waarde

Poging HIMEM te laag te zetten.

FORMULA TOO COMPLEX

Meer dan twee statements van de vorm IF string THEN... zijn uitgevoerd.

OVERFLOW

Het resultaat van een berekening is te groot , om in het BASIC getalformaat weer te geven.

REDIM ' D ARRAY

Nadat een array is geDIMensioneerd wordt een poging gedaan dezelfde array weer te DIMensioneren.

Deze foutmelding treedt vooral op , wanneer het DIM statement voorbij het punt staat waar een arrayelement voor het eerst wordt gebruikt en door het systeem wordt geDIMensioneerd (10 plaatsen per dimensie).

RETURN WITHOUT GOSUB

Een RETURN of POP statement wordt uitgevoerd , zonder dat er een corresponderende GOSUB is uitgevoerd.

STRING TOO LONG

Er is een poging ondernomen , om met een string - concatenatie (= optelling) een string te maken met meer 255 karakters.

BAD SUBSCRIPT

Er wordt naar een arrayelement verwezen , dat buiten , de specificatie van de DIMensionering ligt.

SYNTAX ERROR

Haakjes vergeten in een expressie , niet toegestaan karakter in een regel , verkeerde interpunctie enz.

TYPE MISMATCH

De linkerkant van een toekenning (= assignment) was een numerieke variabele , terwijl de rechterkant een string was of omgekeerd.

Een funktie , die een string - argument verwacht , wordt een numeriek argument meegegeven of omgekeerd.

UNDEF' D STATEMENT

Er wordt een poging gedaan een GOTO , GOSUB of een THEN statement uit te voeren onder verwijzing naar een niet bestaand regelnummer.

UNDEF' D FUNCTION

Er is een FN ingevoerd , waarvan de naam nooit is geDEFinieerd met een DEF FN statement.

4. Gereserveerde woorden

In de onderstaande tabel zijn alle voor APPLESOFT gereserveerde woorden en tekens ondergebracht met daarachter de interne representatie van dat woord of teken (decimaal).

Elk voor APPLESOFT gereserveerd woord neemt in het geheugen 1 byte in beslag , terwijl door gebruiker gedefinieerde namen van variabelen en funkties , expressies e. d. 1 byte per karakter innemen.
& Wordt alleen voor interne doeleinden door APPLESOFT gebruikt en is geen echt APPLESOFT commando. Het veroorzaakt bij gebruik een niet - conditionele sprong naar adres 03F5.

XPLOT is een gereserveerd woord , dat nog geen betekenis heeft.

COLOR , HCOLOR , SCALE , SPEED en ROT worden alleen als gereserveerd woord herkend , als het eerstvolgende niet - spatie karakter = is.

SCRN , SPC en TAB worden alleen herkend , als het eerstvolgende niet spatie - karakter (is.

HIMEM en LOMEM worden alleen herkend , als het eerstvolgende niet - spatie karakter : is.

TU wordt herkend als een gereserveerd woord , als het niet wordt voorafgegaan door een A

ATN wordt alleen herkend als gereserveerd woord , als er geen spatie staat tussen de T en de N.

CHR\$, LEFT\$, MID\$, RIGHT\$ en STR\$ worden alleen als gereserveerd woord herkend , als het \$ teken aanwezig is als eerste niet - spatie karakter.

INTEGER BASIC

ABS	END	LET	PDL	SAVE
AND	FOR	LIST	PEEK	SCRN
ASC	GOSUB	LOAD	PLOT	SGN
AT	GOTO	LOMEM:	POKE	STEP
AUTO	GR	MAN	POP	TAB
CALL	HIMEM:	MOD	PRINT	TEXT
COLOR=	HLIN	NEW	PR#	THEN
CON	IF	NEXT	REM	TO
DEL	IN#	NOT	RETURN	TRACE
DIM	INPUT	NOTRACE	RND	VLIN
DSP	LEN	OR	RUN	VTAB

APPLESOFT

ABS	(212)	HTAB	(150)	REM	(178)
AND	(205)	IF	(173)	RESTORE	(174)
ASC	(230)	IN#	(139)	RESUME	(166)
AT	(197)	INPUT	(132)	RETURN	(177)
ATN	(225)	INT	(211)	RIGHT\$	(233)
CALL	(140)	INVERSE	(158)	RND	(219)
CHR\$	(231)	LEFT\$	(232)	ROT=	(152)
CLEAR	(189)	LEN	(227)	RUN	(172)
COLOR=	(160)	LET	(170)	SAVE	(183)
CONT	(187)	LIST	(188)	SCALE=	(153)
COS	(222)	LOAD	(182)	SCRN((215)
DATA	(131)	LOG	(220)	SGN	(210)
DEF	(184)	LOMEM:	(164)	SHLOAD	(154)
DEL	(133)	MID\$	(234)	SIN	(223)
DIM	(134)	NEW	(191)	SPC((195)
END	(128)	NEXT	(130)	SPEED=	(169)
EXP	(221)	NORMAL	(157)	SQR	(218)
FLASH	(159)	NOT	(198)	STEP	(199)
FN	(194)	NOTRACE	(156)	STOP	(179)
FOR	(129)	ON	(180)	STORE	(168)
FRE	(214)	ONERR	(165)	STR\$	(228)
GET	(190)	OR	(206)	TAB((192)
GOSUB	(176)	PDL	(216)	TAN	(224)
GOTO	(171)	PEEK	(226)	TEXT	(137)
GR	(136)	PLOT	(141)	THEN	(196)
HCOLOR=	(146)	POKE	(185)	TO	(193)
HGR	(145)	POP	(161)	TRACE	(155)
HGR2	(144)	POS	(217)	USR	(213)
HIMEM:	(163)	PRINT	(186)	VAL	(229)
HLIN	(142)	PR#	(138)	VLIN	(143)
HOME	(151)	READ	(135)	VTAB	(162)
HPLOT	(147)	RECALL	(167)	WAIT	(181)
				XDRAW	(149)

DOS

APPEND	CHAIN	INIT	POSITION	SAVE
BLOAD	CLOSE	LOAD	READ	UNLOCK
BRUN	DELETE	LOCK	RENAME	VERIFY
BSAVE	EXEC	OPEN	RUN	WRITE

5. APPLESOFT geheugenmap

0000 - 01FF	Programma geheugen. Niet toegankelijk voor gebruiker.
0200 - 02FF	Keyboard karakter buffer
0300 - 03FF	Beschikbaar voor gebruiker voor korte machinetaal programma' s.
0400 - 07FF	Display gebied page1 voor tekst of grafics.
0800 - XXXX	Beschikbaar voor gebruiker voor BASIC programma' s machinetaalprogramma' s enz. XXXX is het hoogste in het systeem geïnstalleerde RAM adres.
2000 - 3FFF	HGR display page1.
4000 - 5FFF	HGR2 display page2.
C000 - CFFF	Hardware I / O adressen.
D000 - D7FF	Toekomstige ROM uitbreiding
D800 - F7FF	APPLESOFT
E000 - F7FF	APPLE INTEGER BASIC
F800 - FFFF	APPLE systeem monitor

HEXLDC	DECLC	NAME	USE
\$0000-\$00FF	0-255		HARDWARE PAGE ZERO
\$0000-\$0005	0-5		JUMP INSTRUCTIONS TO CONTINUE IN APPLESOFT
\$0000-\$0001	0-1	ROL~ROH	SHEET-16 (16-BIT INTERPRETER) REGISTER RO
\$0000	0	LOC0	MONITOR MEMORY LOCATION 'LOC0'
\$0001	1	LOC1	MONITOR MEMORY LOCATION 'LOC1'
\$000A-\$000C	10-12		LOCN FOR USER FUNCTION'S JUMP INSTRUCTION
\$000D-\$0017	13-23		GENERAL PURPOSE COUNTERS/FLAGS FOR APPLESOFT
\$001A-\$001B	26-27		HI-RES GRAPHICS ON-THE-FLY SHAPE POINTER
\$001A-\$001B	26-27	SHAPEL~SHAPEH	HIRES POINTER TO SHAPE LIST
\$001C	28		HI-RES GRAPHICS ON-THE-FLY COLOR BYTE
\$001C	28	HCOLOR1	HIRES RUNNING COLOR MASK
\$001D	29	COUNTH	HI-RES GRAPHICS HIGH-ORDER BYTE OF STEP COUNT FOR LINE
\$001E-\$001F	30-31	R15L~R15H	SHEET-16 (16-BIT INTERPRETER) REGISTER R15
\$0020-\$004F	32-79		APPLE II SYSTEM MONITOR RESERVED LOCATIONS
\$0020	32	WNDLEFT	SCROLLING WINDOW LEFT SIDE (0-39 OR \$0-\$27)
\$0021	33	WNDWDTH	SCROLLING WINDOW WIDTH (1-40 OR \$1-\$28) (WNDLEFT+WNDWDTH:40)
\$0022	34	WNDTOP	SCROLLING WINDOW TOP LINE (0-23 OR \$0-\$16)
\$0023	35	WNBDM	SCROLLING WINDOW BOTTOM LINE (0-23 OR \$0-\$16) (WNBDM:WNDTOP)
\$0024	36	CH	CURSOR HORIZONTAL POSITION (0-39 OR \$0-\$27)
\$0025	37	CV	CURSOR VERTICAL POSITION (0-23 OR \$0-\$17)
\$0026-\$0027	38-39	QBASL~QBASH	LO-RES GRAPHICS POINTER TO LEFTMOST BYTE OF CUR PLUT LINE
\$0026-\$0027	38-39	HBASL~HBASH	HI-RES GRAPHICS ON-THE-FLY BASE ADDRESS
\$0028-\$0029	40-41	BASL~BASH	MONITOR BASE ADDRESS POINTER
\$002A-\$002B	42-43	BAS2L~BAS2H	MONITOR BASE ADDRESS POINTER 2
\$002C	44	H2	LOW RES COLOR GRAPHICS H2
\$002C	44	LMNEM	MONITOR MEMORY LOCATION 'LMNEM'
\$002C-\$002D	44-45	RTNL~RTNH	MONITOR RETURN POINTER
\$002D	45	V2	LOW-RES COLOR GRAPHICS V2
\$002D	45	RMNEM	MONITOR MEMORY LOCATION 'RMNEM'
\$002D	45	V2	MONITOR MEMORY LOCATION 'V2'
\$002E	46	MASK	LOW-RES COLOR GRAPHICS MASK
\$002E	46	CHKSUM	MONITOR MEMORY LOCATION 'CHKSUM'
\$002E	46	FORMAT	MONITOR & MINIASSEMBLER MEMORY LOCATION 'FORMAT'
\$002F	47	LASTIN	MONITOR MEMORY LOCATION 'LASTIN'
\$002F	47	LENGTH	MONITOR & MINIASSEMBLER MEMORY LOCATION 'LENGTH'
\$002F	47	SIGN	MONITOR MEMORY LOCATION 'SIGN'
\$0030	48	COLOR	LO-RES COLOR GRAPHICS COLOR (FOR PLOT/HLIN/VLIN FUNCTIONS)
\$0030	48	HMASK	HI-RES GRAPHICS HMASK ON-THE-FLY 8-BIT MASK
\$0031	49	MODE	MONITOR & MINIASSEMBLER MEMORY LOCATION 'MODE'
\$0032	50	INVFLG	VIDEO FORMAT CONTROL 255(\$FF)=NORMAL, 127(\$7F)=FLASHING, 63(\$3F)=INV
\$0033	51	PROMPT	PROMPT CHARACTER PRINTED ON GETLN CALL
\$0034	52	YSAV	MONITOR & MINIASSEMBLER MEMORY LOCATION 'YSAV'
\$0035	53	YSAVI	MONITOR MEMORY LOCATION 'YSAVI'
\$0035	53	L	MINIASSEMBLER MEMORY LOCATION 'L'
\$0036-\$0037	54-55	CSWL~CSWH	PROGRAM COUNTER FOR USER EXIT ON COUT ROUTINE (MONITOR)
\$0038-\$0039	56-57	KSWL~KSWH	PROGRAM COUNTER FOR USER EXIT ON KEYIN ROUTINE (MONITOR)
\$003A-\$003B	58-59	PCL~PCH	USER PROGRAM COUNTER SAVED HERE ON BRK TO MONITOR
\$003C	60	XGT	MONITOR MEMORY LOCATION 'XGT'
\$003C	60	XGTNZ	MONITOR MEMORY LOCATION 'XGTNZ'
\$003C-\$003D	60-61	A1L~A1H	MONITOR WORK BYTE PAIR A1
\$003E-\$003F	62-63	A2L~A2H	MONITOR WORK BYTE PAIR A2
\$0040-\$0041	64-65	A3L~A3H	MONITOR WORK BYTER PAIR A3
\$0042-\$0043	66-67	A4L~A4H	MONITOR WORK BYTE PAIR A4
\$0044	68	FMT	MINIASSEMBLER MEMORY LOCATION 'FMT'
\$0044-\$0045	68-69	ASL~ASH	MONITOR WORK BYTE PAIR A5
\$0045	69	ACC	USER AC SAVED HERE ON BRK TO MONITOR
\$0046	70	XREG	USER X-REG SAVED HERE ON BRK TO MONITOR
\$0047	71	YREG	USER Y-REG SAVED HERE ON BRK TO MONITOR
\$0048	72	STATUS	USER P STATUS SAVED HERE ON BRK TO MONITOR
\$0049	73	SPNT	USER STACK POINTER SAVED HERE ON BRK
\$004A-\$004B	74-75	LOMEML~LOMEMH	POINTER TO LOMEM
\$004C-\$004D	76-77	HIMEML~HIMEMH	POINTER TO HIMEM
\$004E-\$004F	78-79	RNDL~RNDH	16-BIT NO. RANDOMIZED WITH EACH KEY ENTRY
\$0050-\$006F	80-97		GENERAL PURPOSE POINTERS FOR APPLESOFT
\$0050-\$0051	80-81	ACL~ACH	MONITOR POINTER 'AC'
\$0050-\$0051	80-81	DXL~DXH	HIRES GRAPHICS DELTA-X FOR HLIN SHAPE
\$0051	81	SHAPEX	HIRES GRAPHICS SHAPE TEMP.
\$0052	82	DY	HIRES GRAPHICS DELTA-Y FOR HLIN SHAPE
\$0052-\$0053	82-83	XTNDL~XTNDH	MONITOR 16-BIT POINTER 'XTND'
\$0053	83	GURNT	HI-RES GRAPHICS QDRNT 2 LSB'S ARE ROTATION QUADRANT FOR DRAW
\$0054	84	EL	HI-RES GRAPHICS ERROR FOR HLIN
\$0054-\$0055	84-85	AUXL~AUXH	MONITOR 16-BIT POINTER 'AUX'
\$0054-\$0055	84-85	EL~EH	HI-RES GRAPHICS ERROR FOR HLIN
\$0055	85	EH	HI-RES GRAPHICS ERROR FOR HLIN
\$0062-\$0066	98-102		RESULT OF LAST MULTIPLY/DIVIDE
\$0067-\$006B	103-104	START PROG PTR	POINTER TO BEGINNING OF PROGRAM NORMALLY \$0B01
\$0067-\$006A	105-106	LOMEM	POINTER TO START OF SIMPLE VARIABLE SPACE
\$006B-\$006C	107-108	ARRAY POINTER	POINTER TO BEGINNING OF ARRAY SPACE
\$006D-\$006E	109-110	FREE SPACE PTR	POINTER TO END OF NUMERIC STORAGE IN USE
\$006F-\$0070	111-112	STRING POINTER	POINTER TO START OF STRING STORAGE. STRINGS TO END OF MEMORY

HEXLOC	DECLOC	NAME	USE
#0083-#0084	131-132	.	POINTER TO THE LAST-USED VARIABLE'S VALUE
#0085-#009C	133-156	.	GENERAL USAGE
#0095		PICK	MONITOR MEMORY LOCATION 'PICK'
#009D-#00A3	157-163	.	MAIN FLOATING-POINT ACCUMULATOR
#00A4	164	.	GENERAL USE IN FLOATING-POINT MATH ROUTINES
#00A5-#00AB	165-171	.	SECONDARY FLOATING-POINT ACCUMULATOR
#00AC-#00AE	172-174	.	GENERAL USAGE FLAGS/POINTERS
#00AF-#00B0	175-176	PROGRAM POINTER	POINTER TO END OF PROGRAM NOT CHANGED BY LOMEM
#00B1	177	.	CHRGOT S/R CALL - GETS NEXT SEQUENTIAL CHR OR TOKEN
#00B1-#00C8	177-200	.	CHRGOT ROUTINE. CALLED WHEN A-S WANTS ANOTHER CHARACTER
#00B7	183	CHRGOT	CHRGOT S/R CALL. CHRGOT INCREMENTS TXTPTR. CHRGOT DOES NOT
#00B8-#00B9	184-185	.	PTR TO LAST CHAR OBTAINED THRU CHRGOT ROUTINE
#00B8-#00B9	184-185	TXTPTR	TXTPTR - POINTS AT NEXT CHAR OR TOKEN FROM PROG (C/A DEC 78)
#00C9-#00CD	201-205	.	RANDOM NUMBER
#00CA-#00CB	202-203	PPL~PPH	BASIC START-OF-PROGRAM POINTER
#00CC-#00CD	204-205	PVL~PVH	BASIC END OF VARIABLES POINTER
#00CE-#00CF	206-207	ACL~ACH	BASIC ACC
#00D0-#00DF	216-223	.	ONERR POINTERS/SCRATCH
#00D0	216	.	POKE 0 TO CLEAR ERROR FLAG
#00DE	222	.	WHEN ERROR OCCURS~ ERROR CODE APPEARS HERE
#00E0-#00E2	224-226	.	HI-RES GRAPHICS X&Y COORDINATES
#00E4	228	.	HI-RES GRAPHICS COLOR BYTE
#00E5-#00E7	229-231	.	GENERAL USAGE FOR HI-RES GRAPHICS
#00E8-#00E9	232-233	.	POINTER TO BEGINNING OF SHAPE TABLE
#00EA	234	.	COLLISION COUNTER FOR HI-RES GRAPHICS
#00F0-#00F3	240-243	.	GENERAL USE FLAGS
#00F3		SIGN	MONITOR & FLOATING POINT ROUTINES MEMORY LOC 'SIGN'
#00F4	244	X2	MONITOR & FLOATING POINT ROUTINES MEMORY LOC 'X2' (EXPONENT 2)
#00F4-#00F8	244-248	.	ONERR POINTERS
#00F5	245	M2	MONITOR & FLOATING POINT ROUTINES MEMORY LOC 'M2' (MANTISSA 2)
#00F7	247	S16PAQ	SWEET-16 MEMORY LOCATION 'S16PAQ'
#00F8	248	X1	MONITOR & FLOATING POINT ROUTINES MEMORY LOC 'X1' (EXPONENT 1)
#00F9	249	M1	MONITOR & FLOATING POINT ROUTINES MEMORY LOC 'M1' (MANTISSA 1)
#00FC	252	E	MONITOR & FLOATING POINT ROUTINES MEMORY LOC 'E'
#0100-#01FF	256-511	.	SUBROUTINE RETURN STACK
#0200	512	IN	MONITOR & MINIASSEMBLER MEMORY LOCATION 'IN'
#0200-#02FF	512-767	.	KEYIN (INPUT) BUFFER
#0300-#03FF	768-1023	.	AREA CLOBBED BY EITHER MASTER OR SLAVE DISKETTE BOOT
#0300-#03F7	768-1015	.	OFTEN FREE SPACE. NOTE COMPETING USES OFTEN FREE SPACE CONSTRAINTS
#0300 #03AF	768-943	.	DECRWTR PRINTER OUTPUT (IF BLOADED FROM DISK)
#0320-#0321	800-801	XOL~XOH	HI-RES GRAPHICS~ PRIOR X-COORD SAVE AFTER HLIN OR HPLOT
#0322	802	YO	HI-RES GRAPHICS YO - MOST RECENT Y-COORDINATE
#0323	803	BXSAV	HI-RES GRAPHICS 'BXSAV'
#0324	804	HCOLOR	HI-RES GRAPHICS COLOR FOR HPLOT~ HPOSN
#0325	805	HNDX	HI-RES GRAPHICS HNDX - ON-THE-FLY BYTE INDEX FROM BASE ADDRESS
#0326	806	HPAQ	POKE 32 FOR HI-RES P01 PLOTTING~ 64 FOR PAGE2
#0326	806	HPAQ	HI-RES GRAPHICS MEM PAGE FOR PLOTTING GRAPHICS #20 FOR P01 ~#40 FOR PG2
#0327	807	SCALE	ON-THE-FLY SCALE FACTOR FOR DRAW~ SHAPE~ MOVE
#0328-#0329	808-809	SHAPXL~SHAPXH	START-OF-SHAPE-TABLE POINTER
#032A	810	COLLSN	COLLISION COUNT FROM DRAW~DRAW1
#03D0	976	.	DOB RE-ENTRY POINT (3D00)
#03D0	976	.	INITIALIZE OR RE-INITIALIZE DOB (3D00)
#03D3	979	.	DOB 3.1 HARD ENTRY POINT
#03D4	982	.	DOB 3.1 ENTRY POINT FOR I/O PACKAGE
#03D9	985	.	DOB 3.1 ENTRY POINT FOR RWTS
#03DC	988	.	DOB 3.1 ENTRY POINT TO LOAD Y~A WITH ADDRESS AT END OF SYS BUFFER
#03E3	995	995	DOB 3.1 ENTRY POINT TO LOAD Y~A WITH ADDRESS OF IOBLK
#03EA	1002	1002	DOB 3.2 ENTRY POINT FOR ROUTINE THAT UPDATES I/O HOOK TABLES
#03FB	1016	USRADR	CTL-Y WILL CAUSE JSR HERE
#03FB	1019	NMI	NMI'S VECTORED TO THIS LOCATION
#03FE	1022	IRGADR	MONITOR MEMORY LOCATION 'IRGADR'
#03FE-#03FF	1022-1023	.	IRQ'S VECTORED TO ADDRESS WHOSE POINTER IS HERE
#0400-#07FF	1024-2043	.	SCREEN BUFFER (HARDWARE PAGES 4-7) (LOW-RES GRAPHICS & TEXT PAGE 1)
#0478+S	1144+S	BRATE	SERIAL INTERFACE BAUD QUANTUM RATE. #1= 19200 BAUD; #40=300 BAUD
#0478+S	1144+S	.	SCRATCHPAD MEMORY BYTE FOR PERIPHERAL IN SLOT #8
#04FB+S	1272+S	STBITS	SERIAL INTERFACE: CONTAIN NUMBER OF STOP BITS (INCLUDING 1 PARITY BIT)
#04FB+S	1272+S	.	SCRATCHPAD MEMORY BYTE FOR PERIPHERAL IN SLOT #8
#0578+S	1400+S	STATUS	SERIAL INTERFACE: PARITY CHECKSUM OPTIONS (SEE MANUAL)
#0578+S	1400+S	.	SCRATCHPAD MEMORY BYTE FOR PERIPHERAL IN SLOT #8
#05FB+S	1528+S	.	SCRATCHPAD MEMORY BYTE FOR PERIPHERAL IN SLOT #8
#0678+S	1656+S	BYTE	SERIAL INTERFACE INPUT OUTPUT BUFFER
#0678+S	1656+S	.	SCRATCHPAD MEMORY BYTE FOR PERIPHERAL IN SLOT #8
#06FB	1784+S	.	SCRATCHPAD MEMORY BYTE FOR PERIPHERAL IN SLOT #8
#06FB+S	1784+S	PHDTH	SERIAL INTERFACE PRINT LINE WIDTH (# CHARS PER LINE)
#0778+S	1912+S	NBITS	SERIAL INTERFACE NUMBER OF DATA BITS PLUS 1 FOR START BIT
#0778+S	1912+S	.	SCRATCHPAD MEMORY BYTE FOR PERIPHERAL IN SLOT #8
#07FB+S	2040+S	FLAGS	SERIAL INTERFACE OPERATION MODE
#07FB+S	2040+S	.	INTERRUPT RETURN MEMORY BYTE FOR PERIPHERAL IN SLOT #8
#0800	2048	.	DEFAULT INTEGER BASIC LOMEM
#0800-#09FF	2048-2559	.	AREA CLOBBED BY EITHER MASTER OR SLAVE DISKETTE BOOT
#0800-#08FF	2048-3071	.	SECONDARY SCREEN BUFFER (TEXT & LOW-RES GRAPHICS PAGE 2)

HEXLOC	DECLOC	NAME	USE
#0800-#C000	2048-49152		RANGE OF POSSIBLE SETTINGS FOR HIMEM (DEPENDING UPON MEM SIZE) DOS
#0800-LOMEM	2048-LOMEM		PROGRAM STORAGE FOR ROM VERSION OF APPLESOFT
#0C00	3072		DEFAULT LOCATION FOR START OF SHAPE TABLE AS SET BY HI-RES SHAPE LOAD
#0C00-#1FFF	3072-8191		OFTEN FREE SPACE
#0CF2	3314		TO CNVRT A/S PROG FM ROM TO CASSETTE LOAD PROG* CALL 3314*LIST*SAVE
#1800-#3FFF	4000-16383		THIS REGION OF MEMORY IS CLOBERBERED BY A SLAVE DISKETTE BOOT
#1800-#4000	4012-16384		RAWDOS (VERSION OF DOS USED WITH MASTER CREATE - FROM DISK)
#2000-#3FFF	8192-16383		HI-RES GRAPHICS PAGE 1
#3000-LOMEM	12288-LOMEM		PROGRAM STORAGE FOR RAM VERSION OF APPLESOFT
#3F3-#3F4	1011-1012		DOS 3.1 - POKE TO ZEROS TO REBOOT HELLO PROGRAM
#4000-#4320	16384-17696		NORMAL LOCATION FOR KAPOR'S HI RES TEXT SET
#4000-#3FFF	16384-24575		HI-RES GRAPHICS PAGE 2
#4500	17664		CALL FOR INVERSION BY KAPOR'S ROUTINE
#4500-4520	17664-17696		S/R W/ KAPOR'S HI-RES TEXT SET TO INVERT WHITE TO BLACK & VICEVERSA
#5600-#B000	22016-72768		DISK OPERATING SYSTEM (DOS3.1)
#5600-#9853	-27136--26541		DOS 3.1 USER BUFFER #1
#5600-#9700	-27136--26880		DOS 3.1 USER BUFFER #1 DATA BUFFER
#5701-#9802	-26879--26622		DOS 3.1 USER BUFFER #1 - LIST OF SECTOR & TRACK NUMBERS USED
#5801-#9853	-26623--26341		DOS 3.1 USER BUFFER #1 - FILE NAME & MISC DATA
#5D10-?	-25328-?		STARTING ADDRESSES FOR VARIOUS DOS3.1 TASKS
#5D73-#A7DF	-25229--25561		SYSTEM SECTION OF DOS 3.1
#5D89	-29159		INITIALIZE OR RE-INITIALIZE DOS
#5E4D	-23011		ROUTINE WHICH HANDLES DOS INPUT HOOK
#5E7E	-24962		ROUTINE WHICH HANDLES DOS OUTPUT HOOK
#A184	-24140		ADDRESS FOR DOS3.1 PR# COMMAND
#A189	-24135		ADDRESS FOR DOS 3.1 IN# COMMAND
#A18E	-24130		ADDRESS FOR DOS 3.1 MON COMMAND
#A1DC	-24100		ADDRESS FOR DOS 3.1 MAXFILES COMMAND
#A1EE	-24082		ADDRESS FOR DOS 3.1 DELETE COMMAND
#A1FC	-24068		ADDRESS FOR DOS 3.1 LOCK COMMAND
#A200	-24064		ADDRESS FOR DOS 3.1 BSAVE COMMAND
#A200	-24064		ADDRESS FOR DOS 3.1 UNLOCK COMMAND
#A208	-24036		ADDRESS FOR DOS 3.1 VERIFY COMMAND
#A20C	-24052		ADDRESS FOR DOS 3.1 RENAME COMMAND
#A223	-24029		ADDRESS FOR DOS 3.1 APPEND COMMAND
#A236	-24010		ADDRESS FOR DOS 3.1 OPEN COMMAND
#A278	-23944		ADDRESS FOR DOS 3.1 CLOSE COMMAND
#A2EC	-23828		ADDRESS FOR DOS 3.1 BLOAD COMMAND
#A327	-23769		ADDRESS FOR DOS 3.1 BRUN COMMAND
#A330	-23760		ADDRESS FOR DOS 3.1 SAVE COMMAND
#A3A5	-23643		ADDRESS FOR DOS 3.1 LOAD COMMAND
#A47A	-23434		ADDRESS FOR DOS 3.1 RUN COMMAND
#A48D	-23411		ADDRESS FOR DOS 3.1 CHAIN COMMAND
#A4A5	-23387		ADDRESS FOR DOS3.1 WRITE COMMAND
#A480	-23376		ADDRESS FOR DOS 3.1 READ COMMAND
#A4E4	-23324		ADDRESS FOR DOS 3.1 INIT COMMAND
#A501	-23295		ADDRESS FOR DOS 3.1 NOMON COMMAND
#A50D	-23283		ADDRESS FOR DOS 3.1 FP COMMAND
#A531	-23247		ADDRESS FOR DOS 3.1 INT COMMAND
#A54F	-23217		ADDRESS FOR DOS 3.1 EXEC COMMAND
#A566	-23210		ADDRESS FOR DOS 3.1 POSITION COMMAND
#A7E0-#A863	-22560--22439		DOS COMMAND TABLE
#A8CD-#A980	-22323--22144		DOS ERROR MSG TABLE
#A976-#A997	-22122--22121		DOS INTERNAL HOOK ADDRESS TO OUTPUT A CHARACTER
#A998-#A999	-22120--22119		DOS INTERNAL HOOK ADDRESS TO INPUT A CHARACTER
#A9A3-#A9A4	-22109--22108		LENGTH OF BLOADED FILE
#A9B5-#A9B6	-22091--22090		STARTING ADDRESS OF BLOADED FILE
#AA0B	-22005		START OF LIST OF POINTERS TO SECTIONS OF DOS 3.1 I/O PACKAGES
#AA3F-#B2CE	-21953--19762		DOS 3.1 I/O PACKAGE
#B3EF-#B642	-19473--18878		DOS 3.1 SYSTEM BUFFER (FOR CATALOG ETC.)
#BDD0	-17152		ROUTINE WHICH READS IN DIRECTORY OFF DISK
#BF6			VOL NO OF CURRENT DISK
#BFFF	-16384		HIGHEST RAM MEMORY ADDRESS
#BFFF	-16384		DEFAULT INTEGER BASIC HIMEM (W/O DOB* 48K MACHINE)
#C000	-16384	KBD ~ IOADR	READ KEYBOARD. IF VAL>127 THEN KEY WAS PRESSED
#C000-#C00F	-16384--16369		KEYBOARD INPUT SUBROUTINE
#C000-#CFFF	-16384--12289		ADDRESSES DEDICATED TO HARDWARE FUNCTION
#C010	-16368	K8DBTB	CLEAR KEYBOARD STROBE. POKE 0 ALWAYS AFTER READING KBD.
#C010-#C01F	-16368--16353		CLEAR KEYBOARD STROBE SUBROUTINE
#C020	-16352	TAPEOUT	MONITOR MEMORY LOCATION 'TAPEOUT'
#C02X	-16352		TOGGLE CASSETTE OUTPUT
#C030	-16336	SPKR	PEEK TO TOGGLE SPEAKER
#C04X	-16320		OUTPUT STROBE TO GAME I/O CONNECTOR
#C050	-16304	TXICLR	POKE TO 0 TO SET GRAPHICS MODE
#C051	-16303	TXTSET	POKE 0 TO SET TEXT MODE
#C052	-16302	MIXCLR	POKE 0 TO SET BOTTOM 4 LINES TO GRAPHICS
#C053	-16301	MIXSET	POKE=0 TO SELECT TEXT/GRAPHICS MIX (BOTTOM 4 LINES TEXT)
#C054	-16300	LOWSCR	*POKE TO 0 TO DISPLAY PRIMARY PAGE (PAGE 1)
#C055	-16299	HISCR	POKE TO 0 TO DISPLAY SECONDARY PAGE (PAGE2)
#C056	-16298	LORES	POKE TO 0 TO SET LO-RES GRAPHICS
#C057	-16297	HIRES	POKE TO 0 TO SET HI-RES GRAPHICS

HEXLOC	DECLOC	NAME	USE
%C058	-16296		POKE 0 TO CLEAR GAME I/O OUTPUT AN0
%C059	-16295		POKE 0 TO SET GAME I/O OUTPUT AN0
%C05A	-16294		POKE 0 TO CLEAR GAME I/O OUTPUT AN1
%C05B	-16293		POKE 0 TO SET GAME I/O OUTPUT AN1
%C05C	-16292		POKE 0 TO CLEAR GAME I/O OUTPUT AN2
%C05D	-16291		POKE 0 TO SET GAME I/O OUTPUT AN2
%C05E	-16290		POKE 0 TO CLEAR GAME I/O OUTPUT AN3
%C05F	-16289		POKE 0 TO SET GAME I/O OUTPUT AN3
%C060	-16288	TAPEIN	MONITOR MEMORY LOCATION 'TAPEIN'
%C060/8	-16288		STATE OF 'CASSETTE DATA IN' APPEARS IN BIT 7
%C061	-16287		PEEK TO READ PDL(0) IF >127 SWITCH ON
%C062	-16286		PEEK TO READ PDL(1) PUSH BUTTON SWITCH
%C063	-16285		PEEK TO READ PDL(2) PUSH BUTTON SWITCH
%C064	-16188	PADDLO	MONITOR MEMORY LOCATION PADDLO
%C064/C	-16188		STATE OF TIMER OUTPUT FOR PADDLE 1 APPEARS IN BIT 7
%C065/D	-16187		STATE OF TIMER OUTPUT FOR PADDLE 1 APPEARS IN BIT 7
%C066/E	-16186		STATE OF TIMER OUTPUT FOR PADDLE 2 APPEARS IN BIT 7
%C067/F	-16185		STATE OF TIMER OUTPUT FOR PADDLE 3 APPEARS IN BIT 7
%C070	-16272	PTRIQ	MONITOR MEMORY LOCATION 'PTRIQ' (PADDLE TRIGGER)
%C07X	-16272	PTRIQ	TRIGGERS PADDLE TIMERS DURING PHI-2
%C08X	-16256		DEVICE SELECT 0
%C09X	-16240		DEVICE SELECT 1
%C0AX	-16224	DEVICE SELECT 2	DEVICE SELECT 2
%C08X	-16208		DEVICE SELECT 3
%C0CX	-16192		DEVICE SELECT 4
%C0DX	-16176		DEVICE SELECT 5
%C0E8	-16152		ADDRESS TO POWER DOWN DISK IN SLOT 6
%C0E9	-16151		ADDRESS TO POWER UP DISK IN SLOT 6
%C0EX	-16160		DEVICE SELECT 6
%C0FX	-16144		DEVICE SELECT 7
%C100	-16128		CALL -16128 IS EQUIVALENT TO PR#1 FOR INITIALIZING SERIAL INTERFACE
%C100	-15842		STANDARD CHARACTER I/O SUBROUTINE ENTRY POINT FOR SLOT #1
%C300	-15616		STANDARD CHARACTER I/O SUBROUTINE ENTRY POINT FOR SLOT #2
%C400	-15360		STANDARD CHARACTER I/O SUBROUTINE ENTRY POINT FOR SLOT #3
%C500	-15104		STANDARD CHARACTER I/O SUBROUTINE ENTRY POINT FOR SLOT #4
%C600	-14848		STANDARD CHARACTER I/O SUBROUTINE ENTRY POINT FOR SLOT #5
%C700	-14592		STANDARD CHARACTER I/O SUBROUTINE ENTRY POINT FOR SLOT #6
%C800-%CFFF	-14336--12289		PIN 20 ON ALL PERIPH CONCTRS GOES LOW DURING PHIO ON READ OR WRITE
%C93D	-14109		SERIAL INTERFACE BATCH INPUT ROUTINE - A1&2 SPECIFY MEMORY RANGE
%C941	-14105		SERIAL INTERFACE BATCH OUTPUT ROUTINE - A1 & A2 SPECIFY MEMORY RANGE
%C500	-16384+256*8		TRANSMIT ASCII CHAR IN ACCUMULATOR OUT VIA SERIAL INTERFACE IN SLOT 8
%D000	-12288	SETHRL	HI-RES GRAPHICS INIT S/R CALL (ROM VERSION)
%D000-%D3FF	-12288--11265		HI-RES GRAPHICS ROM
%D000-%D7FF	-12288--10241		ROM SOCKET D0
%D00E	-12274	HCLR	HI-RES GRAPHICS CLEAR S/R CALL
%D010	-12272	BKGNDO	HI-RES GRAPHICS 'BKGNDO' (HCOLOR1 SET FOR BLACK BKGNDO)
%D012	-12270	BKGNDO	HI-RES GRAPHICS MEMORY LOCATION 'BKGNDO' (ROM)
%D1FC	-11780		HI-RES GRAPHICS FIND S/R CALL: PARAM=SHAPE*ROT*SCALE
%D2F9	-11527		HI-RES GRAPHICS POSN S/R CALL: PARAM=XO*YO*COLR
%D30E	-11506		HI-RES GRAPHICS PLOT S/R CALL: PARAM=XO*YO*COLR
%D314	-11500		HI-RES GRAPHICS LINE S/R CALL: PARAM=XO*YO*COLR
%D331	-11471		HI-RES GRAPHICS BKGNDO S/R CALL: PARAM=COLR
%D337	-11465		HI-RES GRAPHICS LINE S/R CALL: PARAM=XO*YO*COLR
%D33A	-11462		HI-RES GRAPHICS DRAW1 S/R CALL: PARAM=XO*YO*COLR*SHAPE*ROT*SCALE
%D389	-11335		HI-RES GRAPHICS SHLAD S/R CALL
%D48C	-11076		INTEGER BASIC PA#1 APPEND PROGRAM ENTRY
%D4F2	-11022		TO CONVERT A/S FM CASSETTE TO ROM- LD FM CASS*CALL -11022*LIST*SAVE
%D535	-10955		INTEGER BASIC PA#1 TAPE VERIFY PROG ENTRY
%D6DD	-10531		INTEGER BASIC PA#1 RENUMBER PROG ENTRY (WHOLE PROG)
%D6E7	-10521		INTEGER BASIC PA#1 RENUMBER PROG ENTRY (PART PROG)
%D717	-10473		INTEGER BASIC PA#1 MUSIC PROG ENTRY
%D800-%DFFF	-10240--8193		ROM SOCKET D8
%D867	-8867		FRMMUX S/R. EVALS FORMULA EXP. INTO FLOATING PT ACCUM
%DEC9	-8503		SNERR S/R. PRINTS "SYNTAX ERROR" AND HALTS PROG
%E000	-8192	BASIC	ENTER INTEGER BASIC
%E000-%E7FF	-8192--6145		ROM SOCKET E0 (INTEGER BASIC)
%E003	-8189	BASIC2	ENTRY 2 OF INTEGER BASIC
%E368	-7317	MEMFUL	INTEGER BASIC MEMORY FULL ERROR
%E518	-6885		INTEGER BASIC DECIMAL LPRINT S/R
%E6F8	-6408		GETBYT S/R. EVALS FORMULA & CONVS TO 1-BYT VAL IN X REG
%E800-%EFFF	-6144--4097		ROM SOCKET E8 (INTEGER BASIC)
%EEA8	-4504	RNGERR	INTEGER BASIC RANGE ERROR
%F000-%F7FF	-4096--2049		ROM SOCKET F0 (1K INTEGER BASIC~ 1 K MONITOR)
%F11E	-3810	ACADR	HI-RES GRAPHICS 2-BYTE TAPE READ SETUP
%F666	-2458		TURN ON MINIASSEMBLER
%F689	-2423		SWEET-16 INTERPRETER ENTRY
%F800	-2048	PLOT	MONITOR S/R PLOT A POINT (LO-RES) AC:Y-COORD Y:X-COORD
%F800	-2048	PLOT	MONITOR S/R PLOT A POINT. AC:Y-COORD*Y:X-COORD
%F800-%FFFF	-2048--1		ROM SOCKET F8 (MONITOR)

HEXLOC	DECLOC	NAME	USE
%B0C	-2036	RTMASK	MONITOR MEMORY LOCATION 'RTMASK'
%B0E	-2034	PLOT1	MONITOR MEMORY LOCATION 'PLOT1'
%B19	-2023		HLINE S/R (SEE CALL-APPLE NOV/DEC 78 P94)
%B19	-2023	HLINE	MONITOR S/R TO DRAW A HORIZONTAL LINE (LO-RES)
%B1C	-2020	HLINE1	MONITOR MEMORY LOCATION 'HLINE1'
%B26	-2010	VLINEZ	MONITOR MEMORY LOCATION 'VLINEZ'
%B28	-2008	VLINE	DRAW A VERTICAL LINE
%B31	-1999	RTB1	MONITOR MEMORY LOCATION 'RTB1'
%B32	-1998	CLRBCR	CLEAR SCREEN - GRAPHICS MODE
%B32	-1998	CLRBCR	CLEAR LOW RES GRAPHICS SCREEN1
%B36	-1994	CLRTOP	MONITOR MEMORY LOCATION 'CLRTOP'
%B38	-1992	CLRSC2	MONITOR MEMORY LOCATION 'CLRSC2'
%B3C	-1988	CLRSC3	MONITOR MEMORY LOCATION 'CLRSC3'
%B47	-1977	QBASCALC	MONITOR S/R TO CALCULATE GRAPHICS BASE ADDRESS
%B56	-1962	QBALC	MONITOR MEMORY LOCATION 'QBALC'
%B5F	-1953	NXTCOL	MONITOR S/R - INCREMENT COLOR BY 3
%B64	-1948	SETCOL	MONITOR S/R TO ADJUST COLOR BYTE FOR BOTH HALVES EQUAL
%B71	-1935	SCRN	SCRN S/R (LO-RES GRAPHICS) (SEE CALL-APPLE DEC78)
%B71	-1935	SCRN	MONITOR S/R TO GET SCREEN COLOR AC.Y-COORD'Y.X-COORD
%B79	-1927	SCRN2	MONITOR MEMORY LOCATION 'SCRN2'
%B7F	-1921	RTMSKZ	MONITOR MEMORY LOCATION 'RTMSKZ'
%B82	-1918	INSDS1	MONITOR MEMORY LOCATION 'INSDS1'
%B8E	-1906	INSDS2	MONITOR S/R - DISASSEMBLER ENTRY
%B9B	-1893	IEVEN	MONITOR MEMORY LOCATION 'IEVEN'
%BA3	-1883	ERR	MONITOR MEMORY LOCATION 'ERR'
%BA9	-1879	GETFMT	MONITOR MEMORY LOCATION 'GETFMT'
%BBE	-1858	MNNDX1	MONITOR MEMORY LOCATION 'MNNDX1'
%BC2	-1854	MNNDX2	MONITOR MEMORY LOCATION 'MNNDX2'
%BC9	-1847	MNNDX3	MONITOR MEMORY LOCATION 'MNNDX3'
%BD0	-1840	INSTDSP	MONITOR & MINIASSEMBLER MEMORY LOCATION 'INSTDSP'
%BD4	-1836	PRNTOP	MONITOR MEMORY LOCATION 'PRNTOP'
%BD8	-1829	PRNTBL	MONITOR MEMORY LOCATION 'PRNTBL'
%BF3	-1803	PRMN1	MONITOR MEMORY LOCATION 'PRMN1'
%BF9	-1799	PRMN2	MONITOR MEMORY LOCATION 'PRMN2'
%F10	-1776	PRADR1	MONITOR MEMORY LOCATION 'PRADR1'
%F14	-1772	PRADR2	MONITOR MEMORY LOCATION 'PRADR2'
%F26	-1754	PRADR3	MONITOR MEMORY LOCATION 'PRADR3'
%F2A	-1750	PRADR4	MONITOR MEMORY LOCATION 'PRADR4'
%F30	-1744	PRADR5	MONITOR MEMORY LOCATION 'PRADR5'
%F38	-1736	RELADR	MONITOR MEMORY LOCATION 'RELADR'
%F40	-1728	PRNTYX	MONITOR S/R- PRINT CONTENTS OF Y AND X AS 4 HEX DIGITS
%F41	-1727	PRNTAX	MONITOR MEMORY LOCATION 'PRNTAX'
%F44	-1724	PRNTX	MONITOR MEMORY LOCATION 'PRNTX'
%F48	-1720	PRBLNK	MONITOR MEMORY LOCATION 'PRBLNK'
%F4C	-1716	PRBL2	MONITOR S/R- PRINT BLANKS: X REG CONTAINS NUMBER TO PRINT.
%F4C	-1716	PRBL3	MONITOR MEMORY LOCATION 'PRBL3'
%F53	-1709	PCADJ	MINIASSEMBLER MEMORY LOCATION 'PCADJ'
%F54	-1708	PCADJ2	MONITOR & MINIASSEMBLER MEMORY LOCATION 'PCADJ2'
%F56	-1706	PCADJ4	MONITOR MEMORY LOCATION 'PCADJ4'
%F61	-1695	RTB2	MONITOR MEMORY LOCATION 'RTB2'
%F62	-1694	FMT1	MONITOR MEMORY LOCATION 'FMT1'
%F66	-1626	FMT2	MONITOR MEMORY LOCATION 'FMT2'
%F84	-1612	CHAR1	MONITOR & MINIASSEMBLER MEMORY LOCATION 'CHAR1'
%F8A	-1606	CHAR2	MONITOR & MINIASSEMBLER MEMORY LOCATION 'CHAR2'
%F9C	-1600	MNEML	MONITOR & MINIASSEMBLER MEMORY LOCATION 'MNEML'
%FA0	-1536	MNEMR	MONITOR & MINIASSEMBLER MEMORY LOCATION 'MNEMR'
%FA43	-1469	STEP	MONITOR S/R- PERFORM A SINGLE STEP
%FA4E	-1458	XGINIT	MONITOR MEMORY LOCATION 'XGINIT'
%FA78	-1416	XQ1	MONITOR MEMORY LOCATION 'XQ1'
%FA7A	-1414	XQ2	MONITOR MEMORY LOCATION 'XQ2'
%FA86	-1402	IRG	MONITOR S/R- IRG HANDLER
%FA92	-1390	BREAK	MONITOR S/R - BREAK HANDLER
%FA9C	-1380	XBRK	MONITOR MEMORY LOCATION 'XBRK'
%FAA5	-1371	XRTI	MONITOR MEMORY LOCATION 'XRTI'
%FAA9	-1367	XRTB	MONITOR MEMORY LOCATION 'XRTB'
%FAAD	-1363	PCINC2	MONITOR MEMORY LOCATION 'PCINC2'
%FAAF	-1361	PCINC3	MONITOR MEMORY LOCATION 'PCINC3'
%FAB9	-1351	XJSR	MONITOR MEMORY LOCATION 'XJSR'
%FAC4	-1340	XJMP	MONITOR MEMORY LOCATION 'XJMP'
%FAC5	-1339	XJMPAT	MONITOR MEMORY LOCATION 'XJMPAT'
%FACD	-1331	NEWPC1	MONITOR MEMORY LOCATION 'NEWPC1'
%FAD1	-1327	RTNJMP	MONITOR MEMORY LOCATION 'RTNJMP'
%FAD7	-1321	REGDSP	MONITOR S/R TO DISPLAY USER REGISTERS
%FADA	-1318	RODSP1	MONITOR MEMORY LOCATION 'RODSP1'
%FAE4	-1308	RDSP1	MONITOR MEMORY LOCATION 'RDSP1'
%AFD	-1283	BRANCH	MONITOR MEMORY LOCATION 'BRANCH'
%F0B8	-1269	NBRNCH	MONITOR MEMORY LOCATION 'NBRNCH'
%F11	-1263	INITBL	MONITOR MEMORY LOCATION 'INITBL'
%F19	-1255	RTBL	MONITOR MEMORY LOCATION 'RTBL'
%F1E	-1250	PREAD	MONITOR S/R TO READ PADDLE. X-REG CONTAINS PADDLE NUMBER 0-3
%F25	-1243	PREAD2	MONITOR MEMORY LOCATION 'PREAD2'

HEXLOC	DECLOC	NAME	USE
0FB2E	-1234	RTS2D	MONITOR MEMORY LOCATION 'RTS2D'
0FB2F	-1233	INIT	MONITOR S/R- SCREEN INITIALIZATION
0FB39	-1223	SETTXT	MONITOR S/R- SET SCREEN TO TEXT MODE. CLOBBERS ACCUMULATOR
0FB40	-1214	SETGR	MONITOR S/R- SET GRAPHIC MODE (GR). CLOBBERS ACCUMULATOR
0FB48	-1203	SETWND	MONITOR S/R- SET NORMAL WINDOW
0FB58	-1189	TABV	MONITOR MEMORY LOCATION 'TABV'
0FB60	-1184	MULPM	MONITOR MEMORY LOCATION 'MULPM'
0FB63	-1181	MUL	MONITOR S/R- MULTIPLY ROUTINE
0FB65	-1179	MUL2	MONITOR MEMORY LOCATION 'MUL2'
0FB6D	-1171	MUL3	MONITOR MEMORY LOCATION 'MUL3'
0FB76	-1162	MUL4	MONITOR MEMORY LOCATION 'MUL4'
0FB78	-1160	MUL5	MONITOR MEMORY LOCATION 'MUL5'
0FB81	-1151	DIVPM	MONITOR MEMORY LOCATION 'DIVPM'
0FB84	-1148	DIV	MONITOR S/R- DIVIDE ROUTINE
0FB86	-1146	DIV2	MONITOR MEMORY LOCATION 'DIV2'
0FBA0	-1120	DIV3	MONITOR MEMORY LOCATION 'DIV3'
0FBA4	-1116	MD1	MONITOR MEMORY LOCATION 'MD1'
0FBAF	-1105	MD2	MONITOR MEMORY LOCATION 'MD2'
0FBB4	-1100	MD3	MONITOR MEMORY LOCATION 'MD3'
0FBC0	-1088	MDRTS	MONITOR MEMORY LOCATION 'MDRTS'
0FBC1	-1087	BASCALC	MONITOR S/R- CALCULATE TEXT BASE ADDRESS
0FBD0	-1072	BSCLC2	MONITOR MEMORY LOCATION 'BSCLC2'
0FBD9	-1063	BELL1	MONITOR MEMORY LOCATION 'BELL1'
0FBE4	-1052	BELL2	MONITOR S/R- SOUND BELL (BEEPER)
0FBEF	-1041	RTS2B	MONITOR MEMORY LOCATION 'RTS2B'
0FBF0	-1040	STOADV	MONITOR MEMORY LOCATION 'STOADV'
0FBF4	-1036	ADVANCE	MONITOR S/R- MOVE CURSOR RIGHT
0FBFC	-1028	RTS3	MONITOR MEMORY LOCATION 'RTS3'
0FBFD	-1027	VIDOUT	MONITOR S/R- OUTPUT A-REGISTER AS ASCII ON TEXT SCREEN 1
0FC10	-1008	BS	MONITOR S/R TO MOVE CURSOR LEFT (BACKSPACE)
0FC1A	-998	UP ~ CURSUP	MONITOR S/R TO CURSOR UP
0FC22	-990	VTAB	MONITOR S/R- PERFORM A VERTICAL TAB TO ROW SPECIFIED IN ACCUM (#0-#17)
0FC24	-988	VTABZ	MONITOR MEMORY LOCATION 'VTABZ'
0FC28	-981	RTB4	MONITOR MEMORY LOCATION 'RTB4'
0FC2C	-980	ESC1	MONITOR S/R- PERFORM ESCAPE FUNCTIONS
0FC42	-958	CLREOP	MONITOR S/R TO CLEAR FROM CURSOR TO END OF PAGE. CLOBBERS ACC & Y-REG
0FC46	-954	CLEOP1	MONITOR MEMORY LOCATION 'CLEOP1'
0FC58	-936	HOMC	MONITOR S/R TO HOME CURSOR & CLEAR SCREEN. CLOBBERS ACCUM & Y-REG
0FC62	-926	CR	MONITOR S/R TO PERFORM A CARRIAGE RETURN
0FC66	-922	LF	MONITOR S/R TO PERFORM A LINE FEED
0FC70	-912	SCROLL	MONITOR S/R TO SCROLL UP 1 LINE. CLOBBERS ACCUM & Y-REG
0FC76	-906	SCRL1	MONITOR MEMORY LOCATION 'SCRL1'
0FC8C	-884	SCRL2	MONITOR MEMORY LOCATION 'SCRL2'
0FC95	-875	SCRL3	MONITOR MEMORY LOCATION 'SCRL3'
0FC9C	-868	CLEOL	MONITOR S/R TO CLEAR TO END OF LINE
0FC9E	-866	CLEOL2	MONITOR MEMORY LOCATION 'CLEOL2'
0FCA0	-864	CLEOL2	MONITOR MEMORY LOCATION 'CLEOL2'
0FCAB	-856	WAIT	CALL FOR WAIT LOOP
0FCA9	-855	WAIT2	MONITOR MEMORY LOCATION 'WAIT2'
0FCAA	-854	WAIT3	MONITOR MEMORY LOCATION 'WAIT3'
0FCB4	-844	NXTA4	MONITOR S/R TO INCREMENT A4 (16 BITS) THEN DO NXTA1
0FCBA	-838	NXTA1	MONITOR S/R TO INCREMENT A1 (16 BITS). SETT CARRY IF RESULT >=A2.
0FCC8	-824	RTS4B	MONITOR MEMORY LOCATION 'RTS4B'
0FCC9	-823	HEADR	MONITOR MEMORY LOCATION 'HEADR'
0FCD6	-810	WRBIT	MONITOR MEMORY LOCATION 'WRBIT'
0FCD8	-805	ZERDLY	MONITOR MEMORY LOCATION 'ZERDLY'
0FCE2	-798	ONEDLY	MONITOR MEMORY LOCATION 'ONEDLY'
0FCE3	-795	WRTAPE	MONITOR MEMORY LOCATION 'WRTAPE'
0FCEC	-798	RDBYTE	MONITOR MEMORY LOCATION 'RDBYTE'
0FCEE	-786	RDBYT2	MONITOR MEMORY LOCATION 'RDBYT2'
0FCFA	-774	RD2BIT	MONITOR TWO-EDGE TAPE SENSE
0FCFD	-771	RDBIT	MONITOR MEMORY LOCATION 'RDBIT'
0FD0C	-756	RDKEY	GET KEY INPUT FROM THE KEYBOARD. CLOBBERS ACC ~ Y-REG
0FD1B	-741	KEYIN	MONITOR S/R- MONITOR KEYIN ROUTINE
0FD21	-735	KEYIN2	MONITOR MEMORY LOCATION 'KEYIN2'
0FD2F	-721	ESC	MONITOR MEMORY LOCATION 'ESC'
0FD39	-715	RDCCHAR	CALL TO READ KEY & PERFORM ESCAPE FUNCTION IF NECESSARY.
0FD3D	-707	NOTCR	MONITOR MEMORY LOCATION 'NOTCR'
0FD5F	-673	NOTCR1	MONITOR MEMORY LOCATION 'NOTCR1'
0FD62	-670	CANCEL	MONITOR S/R TO PERFORM A LINE CANCEL (\\)
0FD67	-665	GETLNZ	MONITOR S/R TO PERFORM CARRIAGE RETURN AND GET A LINE OF TEXT
0FD6A	-662	GETLN	MONITOR S/R TO GET LINE OF TEXT FROM KEYBD. X RETND W/ # OF CHARS
0FD71	-659	BCKSPC	MONITOR MEMORY LOCATION 'BCKSPC'
0FD75	-651	NXTCHAR	MONITOR MEMORY LOCATION 'NXTCHAR'
0FD7E	-642	CAPTST	MONITOR MEMORY LOCATION 'CAPTST'
0FD80	-640	INSTRDP	MONITOR S/R TO DISASSEMBLE INSTRUCTION AT PCH/PL
0FD84	-636	ADDINP	MONITOR MEMORY LOCATION 'ADDINP'
0FD8E	-626	CROUT	MONITOR S/R TO PRINT A CARRIAGE RETURN. CLOBBERS ACC ~ Y-REG
0FD92	-622	PRA1	MONITOR MEMORY LOCATION 'PRA1'
0FD96	-618	PRYX2	MONITOR MEMORY LOCATION 'PRYX2'

HEXLOC	DECLOC	NAME	USE
%FDA3	-603	XAMB	MONITOR MEMORY LOCATION 'XAMB'
%FDAD	-393	MODBCHK	MONITOR MEMORY LOCATION 'MODBCHK'
%FDB3	-399	XAM	MONITOR MEMORY LOCATION 'XAM'
%FDB6	-586	DATAOUT	MONITOR MEMORY LOCATION 'DATAOUT'
%FDC3	-371	RTS4C	MONITOR MEMORY LOCATION 'RTS4C'
%FDC6	-370	XAMPM	MONITOR MEMORY LOCATION 'XAMPM'
%FDD1	-359	ADD	MONITOR MEMORY LOCATION 'ADD'
%FDDA	-350	PRBYTE	MONITOR S/R TO PRINT CONTENTS OF ACC AS 2 HEX DIGITS
%FDE3	-341	PRHEX	MONITOR S/R TO PRINT A HEX DIGIT
%FDE3	-339	PRHEXZ	MONITOR MEMORY LOCATION 'PRHEXZ'
%FDE3	-339	COUT	MONITOR S/R TO OUTPUT CHAR IN ACC. CLOBBERS ACC~Y-REG~COUT.
%FDF0	-328	COUT1	MONITOR S/R TO GET MONITOR CHARACTER OUTPUT
%FDF6	-322	COUTZ	MONITOR MEMORY LOCATION 'COUTZ'
%FEO0	-312	BL1	MONITOR & MINIASSEMBLER MEMORY LOCATION 'BL1'
%FEO4	-308	BLANK	MONITOR MEMORY LOCATION 'BLANK'
%FEO8	-301	STOR	MONITOR MEMORY LOCATION 'STOR'
%FE17	-489	RTS5	MONITOR MEMORY LOCATION 'RTS5'
%FE18	-488	SETHODE	MONITOR MEMORY LOCATION 'SETHODE'
%FE1D	-483	SETHDZ	MONITOR MEMORY LOCATION 'SETHDZ'
%FE20	-480	LT	MONITOR MEMORY LOCATION 'LT'
%FE22	-478	LT2	MONITOR MEMORY LOCATION 'LT2'
%FE2C	-468	MOVE	MONITOR S/R TO PERFORM A MEMORY MOVE (A1-A2 TO A4)
%FE36	-438	VFY	MONITOR S/R TO PERFORM A MEMORY VERIFY
%FE38	-424	VFYOK	MONITOR MEMORY LOCATION 'VFYOK'
%FE3E	-418	LIST	CALL TO DISASSEMBLE 20 INSTRUCTIONS
%FE63	-413	LIST2	MONITOR MEMORY LOCATION 'LIST2'
%FE78	-392	AIPCLP	MONITOR & MINIASSEMBLER MEMORY LOCATION 'AIPCLP'
%FE7F	-385	AIPCRIS	MONITOR MEMORY LOCATION 'AIPCRIS'
%FE80	-384	SETINV	MONITOR MEMORY LOCATION 'SETINV'
%FE84	-380	SETNORM	MONITOR MEMORY LOCATION 'SETNORM'
%FE86	-378	SETIFLG	MONITOR MEMORY LOCATION 'SETIFLG'
%FE89	-375	SETKBD	MONITOR MEMORY LOCATION 'SETKBD'
%FE8B	-373	INPORT	MONITOR MEMORY LOCATION 'INPORT'
%FE8D	-371	INPRT	MONITOR MEMORY LOCATION 'INPRT'
%FE93	-365	SETVID	MONITOR MEMORY LOCATION 'SETVID'
%FE95	-363	OUTPORT	MONITOR MEMORY LOCATION 'OUTPORT'
%FE97	-361	OUTPRT	MONITOR MEMORY LOCATION 'OUTPRT'
%FE98	-337	IOPT	MONITOR MEMORY LOCATION 'IOPT'
%FEA7	-345	IOPT1	MONITOR MEMORY LOCATION 'IOPT1'
%FEA9	-343	IOPT2	MONITOR MEMORY LOCATION 'IOPT2'
%FEB0	-336	XBASIC	MONITOR S/R TO JUMP TO BASIC
%FEB3	-333	BASCONT	MONITOR S/R TO CONTINUE BASIC
%FEB6	-330	GO	MONITOR MEMORY LOCATION 'GO'
%FEBF	-321	REGZ	MONITOR MEMORY LOCATION 'REGZ'
%FEC2	-318	TRACE	CALL TO PERFORM MONITOR TRACE
%FEC4	-316	STEPZ	MONITOR MEMORY LOCATION 'STEPZ'
%FECA	-310	USR	MONITOR MEMORY LOCATION 'USR'
%FECD	-307	WRITE	MONITOR S/R TO WRITE TO CASSETTE TAPE
%FED4	-300	WR1	MONITOR MEMORY LOCATION 'WR1'
%FEED	-275	WRBYTE	MONITOR MEMORY LOCATION 'WRBYTE'
%FEF	-273	WRBYT2	MONITOR MEMORY LOCATION 'WRBYT2'
%FEF6	-266	CRMON	MONITOR MEMORY LOCATION 'CRMON'
%FEFD	-259	READ	CALL TO READ FROM TAPE - LIMITS A1 & A2
%FF02	-234	READX1	HI-RES GRAPHICS - READ WITHOUT HEADER
%FF0A	-246	RD2	MONITOR MEMORY LOCATION 'RD2'
%FF16	-234	RD3	MONITOR MEMORY LOCATION 'RD3'
%FF20	-211	PRERR	MONITOR S/R TO PRINT "ERR" AND SOUND BELL. CLOBBERS ACC & Y-REG
%FF3A	-198	BELL	MONITOR S/R TO PRINT BELL. CLOBBERS ACC~Y-REG
%FF3A	-198	BELL	CALL HERE TO OUTPUT BELL
%FF3F	-193	RESTORE	MONITOR & SWEET-16 MEMORY LOCATION 'RESTORE'
%FF44	-188	RESTR1	MONITOR MEMORY LOCATION 'RESTR1'
%FF4A	-182	SAVE	MONITOR & SWEET-16 MEMORY LOCATION 'SAVE'
%FF4C	-180	SAV1	MONITOR MEMORY LOCATION 'SAV1'
%FF59	-167	RESET	CALL HERE HAS SAME EFFECT AS PUSHING RESET BUTTON
%FF63	-155	MON	MONITOR S/R- NORMAL ENTRY TO 'TOP' OF MONITOR WHEN RUNNING
%FF69	-151	MONZ	MONITOR S/R TO RESET AND ENTER MONITOR
%FF73	-141	NXTITH	MONITOR MEMORY LOCATION 'NXTITH'
%FF7A	-134	CHRSRCH	MONITOR MEMORY LOCATION 'CHRSRCH'
%FF7C	-132	ZMODE	MONITOR & MINIASSEMBLER MEMORY LOCATION 'ZMODE'
%FF8A	-118	DIG	MONITOR MEMORY LOCATION 'DIG'
%FF90	-112	NXTBIT	MONITOR MEMORY LOCATION 'NXTBIT'
%FF9B	-104	NXTBAS	MONITOR MEMORY LOCATION 'NXTBAS'
%FFA2	-94	NXTBS2	MONITOR MEMORY LOCATION 'NXTBS2'
%FFA7	-89	GETNUM	MONITOR & MINIASSEMBLER MEMORY LOCATION 'GETNUM'
%FFAD	-83	NXTCHR	MONITOR MEMORY LOCATION 'NXTCHR'
%FFBE	-66	TOBUB	MONITOR & MINIASSEMBLER MEMORY LOCATION 'TOBUB'
%FFC7	-57	ZMODE	MONITOR MEMORY LOCATION 'ZMODE'
%FFCC	-52	CHRTBL	MONITOR & MINIASSEMBLER MEMORY LOCATION 'CHRTBL'
%FFEC	-29	SUBTBL	MONITOR MEMORY LOCATION 'SUBTBL'

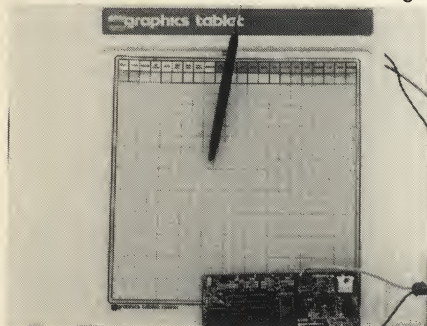
HOOFDSTUK 7 GRAFISCHE VOORSTELLINGEN

1. Grafieken met een grof raster.

De APPLE-II kun je uitstekend gebruiken voor het maken van (architektonische of technische) tekeningen, histogrammen en zelfs originele disco - pop - projecties! Je kunt er je kinderen op laten tekenen en mee laten tekenen...

Het kan niet op !!!!

Er bestaan enige nuttige accessoires, die het professioneel werken met de grafische 'mode' zeer verlevendigen. Zo is er een extra ROM - chip (voor INTEGER - BASIC) te verkrijgen, die ondermeer nuttige grafische routines bevat; er is een handige 'lichtpen' en er is de ongeloofelijke video - set, die het mogelijk maakt foto's, zelfs live - televisiebeelden (persoonsherkenning) te verwerken, en een 'digitizer' - tablet waarmee tekeningen kunnen worden gemaakt.

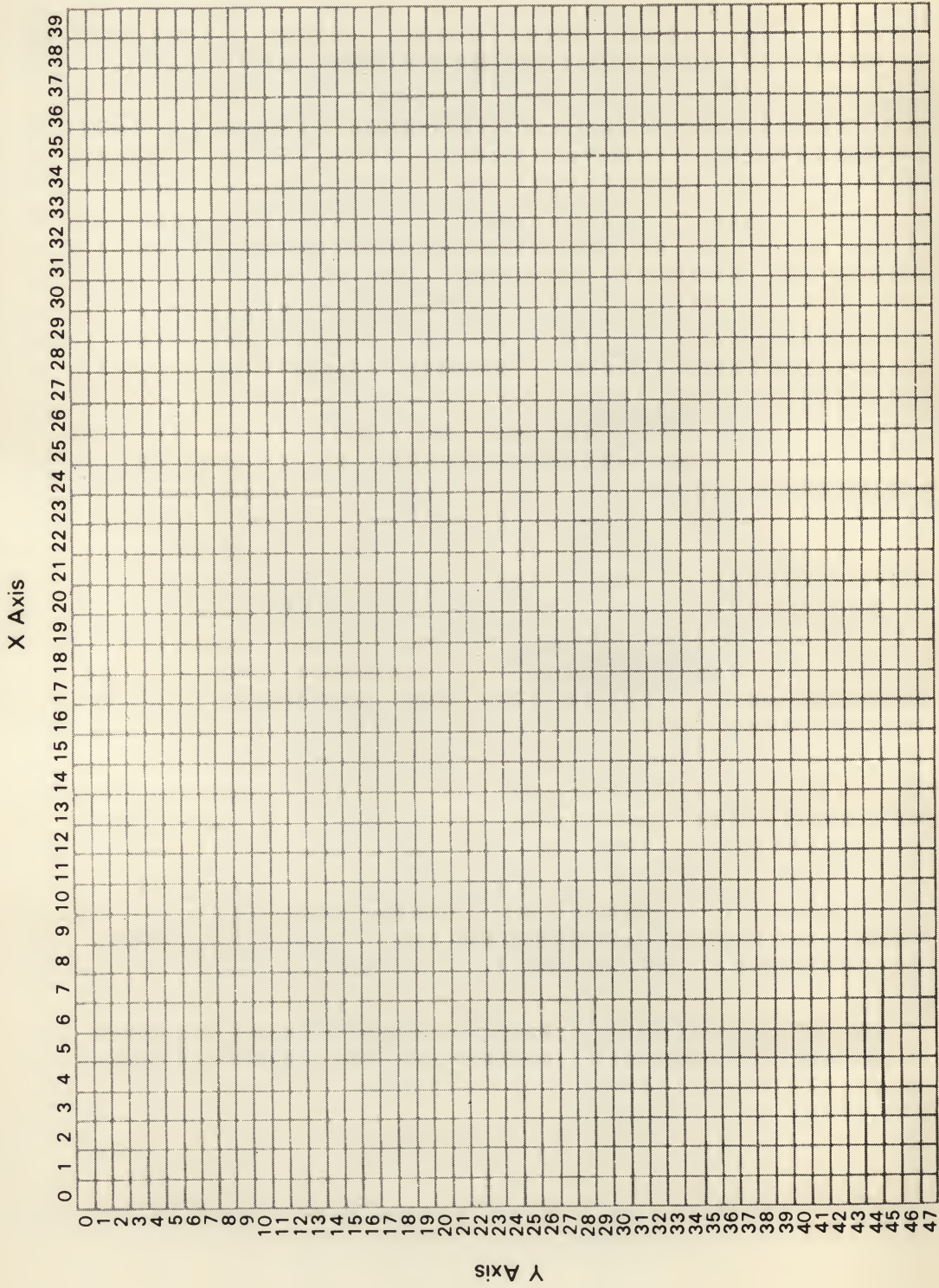


Maar zover behoef je nu echt nog niet te gaan. Wie in kleur wil werken, moet de speciale 'video PALkaart' of de speciale 'RGB - kaart' aanschaffen. Want ook dat is mogelijk: de APPLE-II in kleur.

Nu zijn er twee verschillende grafische 'modes', waarmee we de APPLE-II kunnen laten tekenen. De eerste geeft voorstellingen met een tamelijk grof raster (de 'low resolution graphics') en de tweede geeft voorstellingen met een uiterst fijn raster (de 'high resolution graphics'). Met dat laatste is het mogelijk foto's en t.v. - beelden weer te geven, terwijl een uitgekiend programma tot tekenfilm - achtige effecten aanleiding kan geven. De 'low resolution' grafieken (spreek uit: looh ressoeloesjun) zijn geschikt voor voetbalspelletjes etc. Natuurlijk ook voor histogrammen en andere, grovere professionele toepassingen.

Het venster, waarin getekend kan worden bij deze 'mode', bestaat uit de gehele scherm breedte en de gehele hoogte op 4 regels na, onder in het scherm.

SCREEN LAYOUT FORMS



Het is hierdoor mogelijk programme regels (b.v. een score) zichtbaar te maken op het scherm terwijl er een tekening op het scherm te zien is.

Het raster van deze mode heeft 40 x 40 beeldblokjes.

De oorsprong ligt links boven in het scherm. De coördinaten zijn genummerd van 0 tot 39. Zie de figuur.

Op deze wijze is het scherm ingedeeld in 1600 punten. Elk punt kan aangeduid worden met zijn horizontale en verticale coördinaat , gescheiden door een komma. Zo ligt het punt 0,0 links boven in het scherm , terwijl het punt 39,39 rechts onder te vinden is.

De mogelijkheid is aanwezig , om het gehele scherm als plotvenster te gebruiken.

Hiertoe moet het commando POKE 49234,0 worden gegeven.

Het plotvenster wordt dan 40 x 48 punten groot.

De volgende commando ' s kunnen worden gebruikt , om op het beeldscherm te plotten.

GR imm & def

gr

Geen parameters.

Dit commando brengt de APPLE-II in de ' low resolution graphics ' mode.

Het plotvenster is nu 40 x 40. Het tekstvenster omvat 4 regels onder in het scherm. De cursor wordt verplaatst naar de linker bovenhoek van het tekstvenster , terwijl het scherm geheel gewist wordt.

Met POKE 49234,0 wordt het plotvenster vergroot tot 40 x 48 , terwijl het tekstvenster en de cursor dan geheel verdwijnen.

Wordt daarna een GR commando gegeven , kom je weer in de gemengde tekst - graphics mode.

TEXT imm & def

text

Met dit commando verlaat de APPLE-II de GRafics mode en maakt hij het gehele scherm weer tekstvenster. De cursor gaat naar het begin van de laatste regel op het scherm. Wanneer het tekstvenster oorspronkelijk anders werd gedefinieerd , dan het gehele scherm (zie Hoofdstuk 5 , punt 2) , wordt dit na TEXT automatisch het gehele scherm.

COLOR imm & def

color = aexpr

Dit commando bepaalt de kleur , waarin op het scherm wordt geplot.

Als / aexpr / een real is , wordt die geconverteerd naar integer.

De waarde van aexpr moet liggen tussen 0 en 255 , maar deze waarde wordt modulo 16 verwerkt.

De betekenis van de kleuren is als volgt :

0	zwart
1	magenta
2	donker blauw
3	paars
4	donker groen
5	grijs
6	blauw
7	licht blauw
8	bruin
9	oranje
10	grijs
11	rose
12	groen
13	geel
14	aqua blauw
15	wit

Bij zwart / wit toepassingen zal elke kleur een eigen patroontje op het scherm veroorzaken.

Na GR is COLOR automatisch 0

PLOT imm & def

plot aexpr1,aexpr2

Dit commando plaatst een punt met horizontale coördinaat / aexpr1 / en verticale coördinaat / aexpr2 / op het scherm.

De kleur wordt bepaald door het laatst gegeven COLOR commando.

/ aexpr1 / moet liggen tussen 0 en 40 , / aexpr2 / hoogstens tussen 0 en 47 , anders volgt de foutmelding

?ILLEGAL QUANTITY ERROR

Als geplotted wordt in het gebied 40 t / m 47 , terwijl het systeem in de gemengde tekst - grafics mode is , zal op de plaats , waar anders een punt geplotted zou worden , een karakter verschijnen.
Een karakter neemt de plaats in van twee punten.

HLIN imm & def

hlin aexpr1,aexpr2 at aexpr3

HLIN tekent een horizontale lijn op het scherm van aexpr1,aexpr3 tot aexpr2,aexpr3.
De kleur wordt bepaald door het laatst gegeven COLOR statement.
Voor de grenzen van de expressies gelden dezelfde overwegingen , als bij PLOT.

VLIN imm & def

vlin aexpr1,aexpr2 at aexpr3

Plot een vertikale lijn op het scherm vanaf aexpr3,aexpr1 tot aexpr3,aexpr2.
Dezelfde overwegingen als bij HLIN

SCRN imm & def

scrn(aexpr1,aexpr2)

Geeft de kleurcode van het punt , dat door de expressie wordt gespecificeerd.

In TEXT mode geeft SCRNL een getal terug met de volgende betekenis :

Als / aexpr2 / even is , de minst significante 4 bits van de ASCII waarde van het karakter op het scherm.

Als / aexpr2 / oneven is , de meest significante 4 bits van de ASCII waarde van het karakter op het scherm.

In high resolution mode blijft SCRNL werken in het gebied, dat gereserveerd is voor low resolution grafics.

Met CHR(SCRNL(X-1,2(Y-1))+16*SCRNL(X-1,2(Y-1)+1))$ verkrijgt men ASCII code van het karakter met positie X,Y op het scherm.

2. Grafieken met een fijn raster

Deze grafics mode is ervoor , om zeer gedetailleerde figuren te plotten. Het plotvenster is nu 280 x 160 punten groot , terwijl onder aan het scherm 4 tekstregels over blijven. In deze mode kan het volle scherm voor grafieken benut worden , en wel door het commando POKE 49234,0 uit te voeren.

HGR imm & def

hgr

Dit commando set de High resolution grafics mode. Het plotveld is nu 280 x 160 posities groot , met 4 tekstregels onder aan het scherm. Het scherm wordt zwart en vervolgens wordt HGR page 1 uit het geheugen gedisplayed (2000 - 3FFF hex). HCOLOR verandert niet door dit commando. Gebruik van HGR laat het tekstvenster , zoals het was , zodat alleen de onderste 4 regels worden gedisplayed. Het plotvenster kan vergroot worden tot het gehele scherm door POKE 49234,0. De afmetingen zijn dan 280 x 192 punten. Met HGR komt men weer terug in de gemengde tekst - grafics mode.

Een zeer lang programma kan door HGR gestoord worden. Om dit te voorkomen moet bij systemen met weinig geheugenruimte vooraf HIMEM = 8192 getypt worden , om HGR page 1 te beschermen.

HGR2 imm & def

hgr2

Dit commando brengt het systeem in de high resolution grafics mode , waarbij het plotvenster op volle schermgrootte is. Het scherm wordt zwart en vervolgens wort HGR page 2 uit het geheugen gedisplayed (4000 - 5FFF hex). De tekstpage en HGR page 1 in het geheugen worden niet gewijzigd. Het is dus mogelijk afwisselend HGR page 1 en HGR page 2 op het scherm te krijgen met de respectievelijke commando's: HGR en HGR2. Als het systeem minder dan 24K geheugen heeft , is de HGR2 mode niet toegankelijk. Om te voorkomen , dat programma ' s in HGR page 2 terechtkomen , moet HIMEM op 16384 worden geset.

HCOLOR imm & def

hcolor = aexpr

/ aexpr / moet liggen tussen 0 en 7.
Dit commando bepaalt de kleur , waarin bij HGR , of HGR2
geplot wordt.

0 ZWART
1 GROEN
2 BLAUW
3 WIT
4 ZWART
5 HANGT AF VAN TV
6 HANGT AF VAN TV
7 WIT

HPLLOT imm & def

hplot aexpr1,aexpr2
hplot to aexpr3,aexpr4
hplot aexpr1,aexpr2 to aexpr3,aexpr4

Op de eerste manier gespecificeerd , plot HPLLOT een punt
op de positie met coördinaten / aexpr1 / , / aexpr2 / .
In het tweede geval zal een lijn geplot worden vanaf het
laatst geplotte positie naar / aexpr3 / , / aexpr4 / .
In het derde geval zal een lijn geplot worden vanaf /
aexpr1 / , / aexpr2 / naar / aexpr3 / , / aexpr4 / .
Horizontale coördinaten liggen tussen 0 en 279 ,
vertikale coördinaten tussen 0 en 191.
Als hieraan niet is voldaan , krijg je de foutmelding

?ILLEGAL QUANTITY ERROR

Even tijd voor een korte oefening :

Voer het volgende programma in en RUN het verschillende
malen. Tracht ook enkele wijzigingen aan te brengen en
onderzoek , hoe dit uitwerkt in de programma - executie.

```
10 HGR2
20 HCOLOR=3
30 A=130:B=0:C=75:D=B+40
40 GOSUB 300
50 FOR T = 1 TO 200 : NEXT T
60 B=B+5:D=D+5
70 IF D=190 THEN 90
```



```
80 GOTO 40
90 FOR I=1 TO 300 : NEXT T
100 D=D-5:B=B-5
110 HCOLOR=0
120 GOSUB 300
130 IF B=0 THEN 20
140 GOTO 100
300 HPLOT A,B TO C,D
310 HPLOT A,B TO E,D
320 HPLOT C,D TO E,D
330 RETURN
999 END
```

Probeer met de hiervoor opgedane kennis de positie van de high resolution tekening te veranderen. Hoe stop je de programma - executie ook weer ? Oh ja , CTRL - C , TEXT , LIST.

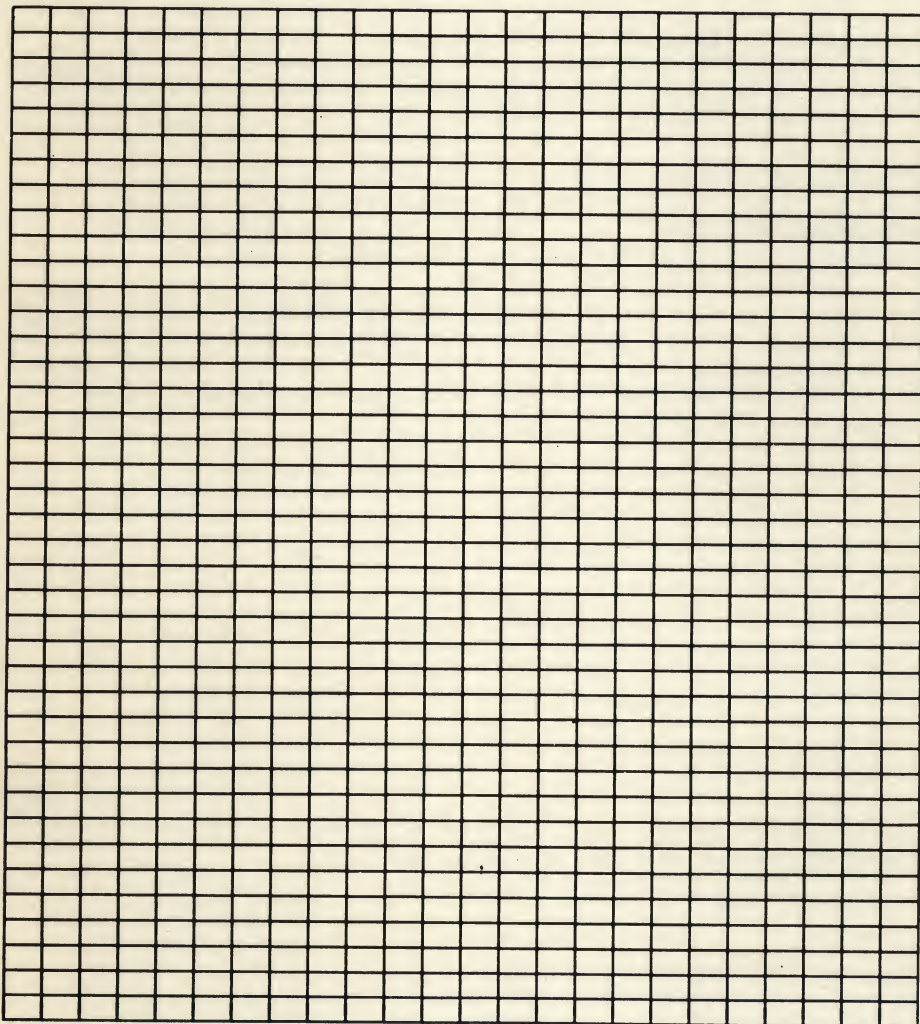
Is dit programma efficient ?

Wat gebeurt er vanaf regel 110 ?

Veel genoeg !

\$2000 8192
 \$2080 8320
 \$2100 8448
 \$2180 8576
 \$2200 8704
 \$2280 8832
 \$2300 8960
 \$2380 9088
 \$2028 8232
 \$20A8 8360
 \$2128 8488
 \$21A8 8616
 \$2228 8744
 \$22A8 8872
 \$2328 9000
 \$23A8 9128
 \$2050 8272
 \$20D0 8400
 \$2150 8528
 \$21D0 8656
 \$2250 8784
 \$22D0 8912
 \$2350 9040
 \$23D0 9168

0 \$00
 1 \$01
 2 \$02
 3 \$03
 4 \$04
 5 \$05
 6 \$06
 7 \$07
 8 \$08
 9 \$09
 10 \$0A
 11 \$0B
 12 \$0C
 13 \$0D
 14 \$0E
 15 \$0F
 16 \$10
 17 \$11
 18 \$12
 19 \$13
 20 \$14
 21 \$15
 22 \$16
 23 \$17
 24 \$18
 25 \$19
 26 \$1A
 27 \$1B
 28 \$1C
 29 \$1D
 30 \$1E
 31 \$1F
 32 \$20
 33 \$21
 34 \$22
 35 \$23
 36 \$24
 37 \$25
 38 \$26
 39 \$27



Map of the High-Resolution Graphics Screen

In each box:

0 \$0000
 1024 \$0400
 2048 \$0800
 3072 \$0C00
 4096 \$1000
 5120 \$1400
 6144 \$1800
 7168 \$1C00

3. Het maken van figuren

APPLESOFI heeft 5 speciale commando 's , waarmee het mogelijk is te manipuleren met figuren in high resolution graphics mode.

Dit zijn DRAW , XDRAW , ROT , SCALE en SHLOAD.

Voordat deze commando 's gebruikt kunnen worden , moet een figuur in het geheugen gedefinieerd worden.

Zo ' n figuur of SHAPE , bestaat uit een aantal plotvectors , die in het geheugen worden opgeslagen.

Een of meerdere figuren vormen samen met indexpointers een z.g. SHAPETABLE (spreek uit : sjeep-teebul).

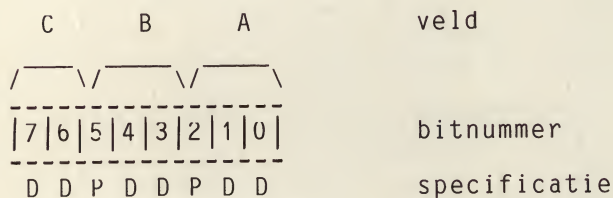
Een SHAPETABLE kan worden opgenomen op cassette , of worden weggeschreven naar floppy disk.

Elk byte in een shape definitie is verdeeld in 3 velden , waarbij elk veld een plotdefinitie bevat.

Een plotdefinitie geeft aan , of er al dan niet geplot moet worden en een verplaatsing naar boven , onder , links of rechts plaats moet vinden.

De commando 's DRAW en XDRAW gaan byte voor byte door de plotdefinities tot ze een byte tegenkomen , die alleen maar nullen bevat.

In figuur A is aangegeven , hoe een shape definitie byte is ingedeeld.



figuur A

Elk paar DD bits geeft een verplaatsing aan , elk P bit of er een verplaatsing geplot moet worden.

De waarden van DD zijn :

DD = 00	een stap omhoog
DD = 01	een stap naar rechts
DD = 10	een stap naar beneden
DD = 11	een stap naar links

Als

P = 1	wordt er geplot
P = 0	wordt er niet geplot

Merk op , dat het veld C geen P bit bevat , zodat hiermee alleen een verplaatsing aangegeven kan worden.

Als veld C en veld B alleen nullen bevat , kan in veld A geen ' stap omhoog zonder plotten ' staan , want dan zou het gehele byte alleen nullen bevatten , waardoor het plotproces wordt afgebroken.

We gaan deze tekening coderen in een tabel. Hierbij kan eventueel gebruik gemaakt worden van een tussenstap, waarbij eerst de pijltjes in een tabel worden geplaatst. Let erop, dat in veld C geen beweging omhoog mag voorkomen, en als veld C nullen bevat, er in veld B geen beweging omhoog zonder plotten mag voorkomen, en dat een byte met enkel nullen niet binnen de plotdefinities mag voorkomen.

veld	C	B	A	C	B	A	vector	code
byte 0		↓	↓		0 1 0	0 1 0	↑	000
1		↔	↔		111	111	→	001 of 01
2		↑	↑		100	000	↓	010 of 10
3	→	↑	↑	0 1	100	100	←	011 of 11
4		↔	↔		101	101		
5		↓	↔		010	101		
6		↓	↓		110	110	↑	100
7		←	↓		011	110	↔	100
8			↔			111	↓	110
9				00	000	000	←	111

Voeg als laatste byte een nulvector toe, om de tabel correct af te sluiten.

Op alle niet gebruikte bit - posities moet nog een 0 geplaatst worden.

Vervolgens moet de tabel naar hexadecimale waarden worden vertaald, om hem te kunnen invoeren in de APPLE - II.

veld	C	B	A	hex	codes
byte 0	0001	0010	= 12	binair	hex
1	0011	1111	= 3F	0000	0
2	0010	0000	= 20	0001	1
3	0110	0100	= 64	0010	2
4	0010	1101	= 2D	0011	3
5	0001	0101	= 15	0100	4
6	0011	0110	= 36	0101	5
7	0001	1110	= 1E	0110	6
8	0000	0111	= 07	0111	7
9	0000	0000	= 00	1000	8
				1001	9
				1010	A
				1011	B
				1100	C
				1101	D
				1110	E
				1111	F

Er is echter nog meer informatie nodig, om de shapetable te completeren. Er moeten nog indexpointers aan toegevoegd worden.

In het geval van ons voorbeeld is dat niet zo moeilijk, omdat de tabel slechts een figuur bevat.

Vervolgens komen er een of meerdere pointers, die verwijzen naar het begin van de verschillende figuren in de tabel. Een pointer is twee bytes groot en bevat het aantal bytes, dat het begin van een figuur van S verwijderd ligt.

In de figuren H en K is de opbouw van een shapetable nogmaals geschetst voor het algemene gebruik en voor het behandelde voorbeeld in het bijzonder.

figuur K

De tabel kan nu in de APPLE-II worden ingevoerd.
We raden aan , als je nog nooit met monitor - commando's hebt kennism gemaakt , Hoofdstuk 8 te bestuderen.

Kies eerst een startadres. Dit adres moet natuurlijk kleiner zijn , dan het grootste in je APPLE-II voorkomende RAM adres , en niet in een gebied komen , dat gebruikt wordt door HGR of HGR2.

Laten we als voorbeeld lokatie 1DFC kiezen. Dit is net onder HGR page 1.

Bedien de RESET toets om in Monitor te komen (of CALL -151).

Tik het startadres 1DFC in , gevolgd door :. Daarna gescheiden door spaties de hexadecimale bytes van de shapetable. Dus :

1DFC: 01 00 04 12 3F 20 64 2D 15 36 1E 07 00(RETURN)

Display de gebruikte geheugenplaatsen , om te controleren of er geen fouten zijn gemaakt.

Gebruik hiervoor bijvoorbeeld :

1DFC.1E09 (Return)

Vervolgens moet in speciaal daarvoor gereserveerde geheugenplaatsen het startadres van de tabel geplaatst worden , zodat APPLESOFT de shapetable kan vinden.

Deze geheugenlokatie zijn de adressen E8 en E9. E8 bevat het minst significante deel van het adres , E9 het meest significante deel.

In het geval van het voorbeeld moet je de plaatsen E8 en E9 vullen met FC en 1D. Om de shapetable te beschermen kun je HIMEM het beste op het startadres van de tabel zetten. Dus :

E8: FC 1D (Return) zet de pointer op adres shapetable

73: FC 1D (Return) zet HIMEM op startadres shapetable

Sla de lengte van de tabel op in adres 0 en 1

0: 0D 00 (Return)

Schrijf de lengte van de shapetable en de shapetable zelf naar cassette (zorg ervoor dat de recorder opneemt en loopt , voordat je de RETURN toets bedient) :

0.1W 1DFC.1E09W (Return)

Aan het begin en einde van de opname hoor je een ' biep ' toontje.

Met de volgende commando ' s kan je shapes manipuleren.

DRAW imm & def

draw aexpr1 f at aexpr2,aexpr3 ij

Als aexpr1 alleen gespecificeerd is , wordt figuur / aexpr1 / geplot op de plaats , waar het laatst een high resolution plot heeft plaats gevonden.

/ aexpr1 / mag natuurlijk niet groter zijn dan in het eerste byte van de shapetable is opgegeven.

Als 3 expressies worden gespecificeerd , dan wordt figuur / aexpr1 / geplot op lokatie / aexpr2 / , / aexpr 3 / van het scherm.

In het geval van het voorbeeld :

DRAW 1 AT 100,100

zal het gedefinieerde figuurtje op positie 100,100 getekend worden.

XDRAW imm & def

xdraw aexpr1 f at aexpr2,aexpr3 ij

XDRAW doet hetzelfde als DRAW met dit verschil , dat elk te plotten punt op het scherm wordt geplot in de kleur , die het complement is van de kleur , die dit punt al op het scherm heeft.

Het is hierdoor mogelijk figuren uit een plaatje weg te halen , zonder de rest van het plaatje te verstoren.

ROT imm & def

rot = aexpr

Met ROT kan de ruimtelijke afstand van de te plotten figuren worden aangegeven.

/ aexpr / moet liggen tussen 0 en 255.

ROT = 0 geeft geen rotatie van de figuur

ROT = 16 geeft een hoekverdraaiing over 90 graden met de wijzer van de klok mee.

ROT = 32 geeft hoekverdraaiing over 180 graden

ROT = 48 geeft hoekverdraaiing over 270 graden

ROT = 64 gelijk aan ROT = 0.

Als SCALE = 1 kunnen slechts 4 mogelijke rotatiehoeken worden opgegeven : 0 , 16, 32 en 48.
Als SCALE = 2 kunnen 8 rotatiehoeken worden opgegeven : 0 , 8 , 16 , 24 , 32 , 40 , 48 , 56.
Als SCALE = 3 kunnen 16 mogelijke verdraaiingen worden opgegeven enz.

Niet herkende rotaties hebben meestal tot gevolg , dat de naastliggende , kleinere rotatie wordt uitgevoerd.

SCALE imm & def

scale = aexpr

Maakt het mogelijk figuren te vergroten.
/ aexpr / moet liggen tussen 0 en 255.

SCALE = 0 vergroot 256 maal.
SCALE = 1 kopieert de figuur zonder vergroting , enz.

SHLOAD imm & def

Dit commando laadt een shapetable van cassette.
De shapetable wordt direct onder HIMEM geplaatst , terwijl na het laden HIMEM wordt verzet naar het begin van de shapetable. Ook de tablepointer wordt op de juiste waarde gezet (E8-E9).
Alleen een RESET onderbreekt een SHLOAD.
De volgende 3 dingen dienen bekend te zijn , om een shapetable naar cassette te kunnen schrijven :

1. startadres van de shapetable
2. laatste adres van de shapetable
3. de lengte van de shapetable in bytes

De lengte van de shapetable moet opgeslagen worden in lokaties 0 en 1 van het geheugen. Minst significante byte in 0 , meest significante byte in 1.

Er kan nu met het monitor W(rite) - commando naar cassette geschreven worden , eerst de lokaties 0 en 1 en dan de shapetable zelf.

In het geval van het hiervoor behandelde voorbeeld :

1. 1DFC
2. 1E09
3. 000D


```

1  REM *****
2  REM * SHAPE CREATION PROGRAM *
3  REM *
4  REM *****
10 DIM S1(100),V1(100)
20 I = 0
30 PRINT "CREATE SHAPE VECTORS"
40 PRINT
41 REM ENTER PLOT ACTIONS
50 V = I: GOSUB 270
58 REM CONTINUE ENTRY UNTIL M$
59 REM EQUALS TERMINAL VALUE "E"
60 IF M$ < > "E" THEN S1(I) = M:I = I + 1: GOTO 50
70 PRINT
71 REM ALLOW CORRECTIONS
80 INPUT "VECTOR TO CHANGE (0=END):";V
90 IF V > 0 THEN V = V - 1: GOSUB 270:S1(V) = M: GOTO 80
99 REM PACK VECTORS INTO V1() ARRAY
100 FOR V = 0 TO I
110 IF B = 2 AND S1(V) > 0 AND S1(V) < 4 THEN 140
120 IF B < 2 AND (S1(V) > 0 OR S1(V) > 4) THEN 140
130 B = 0:Q = Q + 1
140 V1(Q) = V1(Q) + S1(V) * (8 ^ B)
150 B = B + 1
160 IF B > 2 THEN B = 0:Q = Q + 1
170 NEXT V
178 REM DISPLAY THE VECTORS AS
179 REM HEXADECIMAL NUMBERS
180 PRINT "BYTE","VECTOR"
190 FOR V = 0 TO Q
200 H% = V1(V) / 16
210 L% = V1(V) - H% * 16
220 IF H% > 10 THEN H% = H% + 7
230 IF L% > 10 THEN L% = L% + 7
240 PRINT V, CHR$ (H% + 176); CHR$ (L% + 176)
250 NEXT V
260 END
269 REM VECTOR INPUT SUBROUTINE
270 PRINT "VECTOR ";V + 1;": ";
280 INPUT "MOVE: U/D/L/R? ";M$
290 M = 0
300 IF M$ = "R" THEN M = 1
310 IF M$ = "D" THEN M = 2
320 IF M$ = "L" THEN M = 3
330 IF M$ = "E" THEN RETURN
340 INPUT "PLOT (Y=YES,N=NO)? ";P$
350 IF P$ = "Y" THEN M = M + 4: RETURN
360 IF P$ = "N" THEN RETURN
370 GOTO 340

```


HOOFDSTUK 8 MONITOR

Inleiding

De APPLE II computer beschikt over een monitor-hulpprogramma. Dit monitorprogramma bestaat uit een hoofdprogramma met een aantal kleine, maar krachtige subroutines. In feite is de monitor een controleprogramma, dat allerlei processen in de computer reguleert. Na het inschakelen van de computer zorgt een z.g. "bootstrap" programma ervoor, dat er enkele inleidende monitor-routines worden gebruikt, waardoor hij zich direkt gebruiks-klaar aanmeldt. Het bootstrapprogramma ligt onveranderlijk vast en reageert enkel op de voedingsspanning.

Om in de monitor te komen zijn er twee mogelijkheden. De standaard versie van de APPLE II, die niet is uitgerust met een Autostart ROM en niet standaard van Applesoft is voorzien, is hierbij enigszins in het voordeel. De APPLE II PLUS, of EURO-APPLE, met Autostart ROM en standaard Applesoft vraagt om een iets afwijkende benadering. Bovendien beschikt dit type niet standaard over een Mini-Assembler, waarover later meer.

Om vanuit Basic over te schakelen naar de monitor tik je:

```
]CALL-151 (return)
```

De prompt verandert nu in het asteriksteken: *.
Wij kunnen nu volgens 3 methoden te werk gaan, om nader via de monitor geheugenplaatsen uit te lezen.

1. bekijken van een enkel adres
2. bekijken van een geheugenwoord, of regel
3. bekijken van een geheugenblok

Bij de bespreking van de verschillende monitorcommando's betekent "adres" een hexadecimaal getal van 4 cijfers. Daarvan mogen de niet significante nullen worden weggelaten. Dus: 00 01 mag worden genoteerd als 1.
Onder "data" wordt verstaan een hexadecimaal getal van 2 cijfers, waarvan de niet significante nullen eveneens mogen worden weg gelaten. De commando's worden telkens afgesloten met een tik(je) op de RETURN-toets. Daaraan zullen we dus verder voorbijgaan!

1. ENKELE OBSERVATIES

De APPLE II systeemmonitor omvat het bovenste gebied van de geheugenmap van de computer. De erin ondergebrachte routines worden ondermeer uitgebreid beschreven in het APPLE REFERENCE MANUAL, dat bij de computer wordt meegeleverd. Toch zijn de handige programmaroutines lang niet voor iedereen toegankelijk.

Vaak doe je er zelfs ongemerkt een beroep op. Nuttige POKE's en CALL's behoren ertoe. Bijv. CALL-936 is een verwijzing naar een monitorroutine, die het beeldscherm "wist" en de cursor links bovenin plaatst. CALL-936 komt overeen met het Applesoft HOME commando en verwijst naar de hexadecimale lokatie \$FC58.

Het werken met de systeem-monitor brengt je heel dicht bij de ware taal van de 6502-microprocessor, waarmee de APPLE II is uitgerust. Wat nog niet wil zeggen, dat je in de monitor op dezelfde manier te werk kunt gaan, als weleens in mooie assembler listings wordt voorgespiegeld. Werken in de monitor is priegelwerk en het veronderstelt begrip van het hexadecimale en tweetallige getalsstelsel. Aanbevolen litteratuur is: APPLE MACHINE LANGUAGE door Don en Kurt Inman; THE APPLE II MONITOR PEELED door W.Dougherty en 6502 ASSEMBLY LANGUAGE PROGRAMMING door L.A.Leventhal. De boeken werden in opklimmende moeilijkheidsgraad opgenomen en zijn verkrijgbaar bij de boekhandel.

2. DE MONITOR IN EN UIT

Ondermeer bij de behandeling van het DOS kom je een stukje monitor-programmering tegen. Nu is het begrip "monitor-programmering" wat omslachtig en eigenlijk minder juist. De systeemmonitor ligt als programmeerhulp in ROM vast en hangt ten nauwste samen met de BASIC-Interpreter, die ook in ROM werd opgeslagen. Daaraan valt op zichzelf weinig te veranderen. We bedoelen eigenlijk het in een machinetaal programmeren van onze fijne APPLE computer.

Om in de monitor te komen tikken we CALL-151. Heb je de tekst goed gevolgd, dan is je computer al in de monitor en zie je een * als prompt met daarnaast de knipperend cursor.

Om weer terug te komen in Basic tik je CTRL-C (return) en alles is weer bij het oude en nu al wel bekende. Maar verandert een Basic programma nu, wanneer je de monitor in en uit gaat ?

Laten we dit eerst testen:

```
]10 REM MON TEST
20 PRINT 10*10
30 END
```

Let op:]CALL-151
*

Tik nu: CTRL-C (en tik op de returntoets)
Gelukt ?

1 ja, want de Applesoftprompt is terug

Test nu RUN en LIST.

CTRL-C werkt prima met Integer Basic en Applesoft in ROM. Maar het werkt niet met Applesoft vanaf cassette en diskette. De terugkeer wordt dan bewerkstelligd door:

- a) Diskette: *3DOG (return)
- b) Cassette: * OG (return)

Beide instructies grijpen terug op BRANCH instructies van de monitor. Waarover later meer.

Er is evenwel nog een mogelijkheid: druk een keer op de RESET toets. Degenen met een Autostart ROM zijn dan direkt terug in Basic. Heb je geen Autostart ROM dan blijft de prompt een asteriks *.

Nu een andere, veel gevraagde oefening:
Eerst maken we ons beeldscherm schoon: HOME en wissen we ons oude programma: NEW.
Tik nu:

```
]10 PRINT 1
20 REM NAAM
```

Zo'n programma blijft natuurlijk ergens in het geheugen van de APPLE opgeslagen. Maar waar en kunnen we het daar eens bekijken? Tik daarom alvast CALL-151.
 Applesoftprogramma's beginnen onder in het vrije RAM werkgeheugen en volgens de geheugenmap vanaf adres 2048 of \$800. Vanuit de monitor kunnen we dit en de erop volgende adressen inspecteren. Voer dan nu het getal 800 in. De monitor accepteert alleen de hexadecimale adresnotatie - dus: met 16 vingers tellen, zo van: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F,10 etc. Je herinnert je, dat het getal 10 een combinatie is van 1 en 0....
 Applesoftprogramma's beginnen in tegenstelling tot Integer programma's onder in het werkgeheugen.

*800 return

Op de display verschijnt: *800: 00 , hetgeen de 'waarde' is van de 8 bits code, die er vermeld staat. De tweetallige code, die elk bit van een 8 bits (=1 byte) instructie bevat, kan of nul of een (=0 of 1) zijn, maar 1 byte kan een geweldige hoeveelheid combinaties hiervan te zien geven. Maximaal zijn er 255 of \$FF combinaties te bedenken. Door nu 2 bytes (16 bits) te combineren, kan de 6502 microprocessor veel grotere waarden aan. En dat is dan ook precies de manier, waarop hij werkt. Een 2-bytes instructie bestaat uit een Minst Significante Byte en een Meest Significante Byte. Eerst komt de Minst gevalideerde byte, daarna de Meest gewichtige byte. Omdat dit soms moeilijk wordt gevonden eerst een samenvatting:

- 1 Binaire digiT (BIT) kan 0 of 1 zijn
 - 1 BYTE bestaat uit 8 Bits 0000 0000
 - 2 BYTE combinaties bestaan uit een LSB en een MSB
- margin12
 LSB=least significant byte / minste gewicht
 MSB=most significant byte / meeste gewicht

Terug naar de computer interpretatie van ons Basic programma.

Wanneer we opnieuw RETURN geven, verschijnen meer bytes op het scherm.

```
*800
00
08 08 0A 00 BA 31 00    geef nu weer return
*808:13 08 14 00 B2 20 4E 41
```

Wat je bij adres *808 ziet, noemt men een geheugen-WOORD. Een groep van 8 bytes. Alle woorden in het geheugen beginnen op een adres, waarvan het getal door 8 deelbaar is. Denk erom, dat het woord *808 als volgt wordt verdergeteld:

*808 809 80A 80B 80C 80D 80E 80F, dan volgt 810

Hoe spreek je dat uit ? Bijv. 80E - acht nul eeh - that's all !!

Wanneer we de ASCII-code tabel erbij nemen (je vindt hem ook in het Reference Manual op blz.15), ontdek je, dat alle karakters in de computerwereld een hexadecimale 'waarde' vertegenwoordigen. Zo blijkt het getal 10 van ons eerste regelnummer \$0A te zijn. Het \$-teken geeft een hex-waarde aan, dit nogmaals en ten overvloede. Beginnen we daarom vanaf adres \$800 te lezen, dan zoeken we naar de byte, die \$0A is. Op \$803 vinden we hem, gevolgd door twee nullen op \$804. De APPLE heeft 65535 ($16^4 - 1$) adreslokaties als gevolg van de 2-bytes codeafhandeling door de 6502 processor. Met andere woorden zien we op \$803/804 een demonstratie van de LSB en MSB in hun logische opeenvolging. De code BA op adres \$805 staat voor het PRINT-statement! Alle Basic statements hebben hun eigen "byte"-code, hetgeen veel geheugenruimte bespaart.

De waarde 31 is de ASCII-code voor het getal 1. Het einde van een Basic programmaregel is in de monitor een 00, die we dan ook zien op adres \$807. Tikken we nogmaals RETURN om een extra woord in te lezen. De volgende programmaregel in Basic was 20. In het zestientallig stelsel is dat \$14: ($1 \text{ zestiental} + 4$). Op \$80A vinden we het terug. Gevolgd door de MSB: 00.

Nu komt een grapje, dat sommigen weleens in vertwijfeling heeft gebracht. Stel je nu eens voor, dat wij het regelnummer in de monitor wijzigen !?!

Om een adres te wijzigen tikken we het adres, gevolgd door een dubbele punt- dus: 80A:

Wat zal er straks in ons Basic programma te zien zijn, wanneer we \$80A:FF maken ? DOEN!

Nu keren we na een diepgaande omzwerving in het monitor/programma gebeuren terug naar Applesoft. Tik eerst RETURN, daarna 3D0G.

LIST het programma.

Wat zal er gebeuren, wanneer je ook de MSB de waarde \$FF geeft?

CALL-151 en duik terug in de monitor. Tik *800L (L van List).

Dat gaat snel. We zien in een beurt een heel geheugenblok!

Tik nu: *80B:FF en let er tegelijk op, dat de End-Of-Program Pointer een reeks van 3 nullen is. De EOP pointer vind je ook terug op adres 175 en 176, ofwel \$AF en \$B0; in LSB/MSB notatie.

Gemakkelijk, als je het einde van een lang programma in Applesoft wilt zoeken.

Keer weer terug in Applesoft: *3D0G. LIST opnieuw je programma.

Probeer het hoge regelnummer nu te verwijderen. Leuke copyright-bescherming, nietwaar en ssssstt... niet tegen vrienden vertellen natuurlijk!

Zo krijgen we de smaak te pakken. Misschien begrijp je tegelijk, dat we iets beter de weg kunnen vinden in de monitor notatie van ons Basic-programma. Willen we bij een ander programma "binnen kijken", dan zijn hier een paar ASCII-codes voor Applesoft statements:

REM	B2	PRINT	BA	NEW	BF
END	80	GOTO	AB	GET	BE
1-9	31-39			A-Z	41-5A

3. GEHEUGENCOMMANDO'S

Display geheugen

adres	toon inhoud geheugenplaats/adres
adres1.adres2	toon inhoud geheugenblok
return	toon volgende geheugenwoord
.adres	toon geheugeninhoud vanaf laatst getoonde adres tot aangegeven adres

Wijzig geheugen

adres:data1 data2 etc	wijzig geheugeninhoud vanaf adres met opgegeven data (spatie),data etc.
-----------------------	---

Verplaats geheugenblok

adres1µadres2.adres3M	plaats datablok adres 2.adres3 in blok te beginnen bij adres1
-----------------------	--

Vergelijk geheugenblok

adres1µadres2.adres3V	vergelijk blok adres2.adres3 met blok dat begint bij adres1
-----------------------	--

Input/Output commando's

(getal)CTRL-P	stuur informatie naar Slot (getal)
(getal)CTRL-K	Getal tussen 0 en 7. 0 is CRT/TV input gelezen van Slot (getal)

4. DISPLAY COMMANDO'S

Een erg aantrekkelijk monitorgebied is het TEXT-display. De TV-monitor vormt hierbij het venster, waardoor we kunnen zien, wat zich in zeer specifieke geheugenlokaties afspeelt. TEXT-display is een overzichtelijk proces, dat we zowel vanuit het Applesoft programma als het monitorprogramma kunnen besturen. Het TEXT-venster omvat een geheugengebied, dat begint bij adres 1024, of \$400 en dat loopt tot 2047, \$/FF. Men noemt dit gebied het TEXT-venster no. 1, of in computerbargoens TEXT-Page One. Een tweede pagina omvat de volgende 1024 bytes en loopt van \$800 tot \$BFF (2048-3071 decimaal).

Wie rekenen kan zal onmiddellijk opmerken, dat er 1024 bytes ter beschikking zijn, om een gebied van 40 karakters op 24 regels te vullen - we houden dus wat over.

De resterende bytes worden gebruikt als een soort bewaarplaats voor data afkomstig van PROM's op insteekkaarten in de APPLE II Slots. Zowel de Low Resolution Graphics als de tekstkarakters maken gebruik van het TEXT-venster. Zij het in een ander "formaat" - m.a.w. de schrijfwijze is afwijkend. Laten we ons bepalen tot tekstmanipulaties.

Om een letterteken te displayen zullen we rekening moeten houden met een samenhangende reeks bijzonderheden. Zo heb je bij de behandeling van de Applesoft statements al gehoord van de afmetingen van het TEXT-venster en heb je een overzicht gezien van de setting van de bovenkant, de zijkanten en de onderkant van de "scroll-window". POKE 32 t/m 35 bepalen de omvang van het beeldvenster en dat kan maximaal 960 tekens bevatten. Wanneer we na een CALL-151 de adressen \$20/\$23 bekijken, hebben we een ander zicht op de omvang van het beeldvenster. Adres \$32 heeft te maken met de "mode", waarin zich de karakter-output op het tv-scherm bevindt. POKE 50,63 zorgde er immers voor, dat tekst als zwarte tekens op een lichte achtergrond werd geprojecteerd. POKE 50,127 zorgde voor knipperende tekens en POKE 50,255 maakte, dat de tekens weer "normaal" wit-op-zwart werden gezet. Adres 50, \$32, draagt er dus aan bij, dat de output van karakters "gemaskeerd" wordt tot normale, inverse en knipperende tekens. De APPLE II bezit mede daardoor een eigenaardige ASCII Screen Code, waardoor het mogelijk is, dat zowel CHR\$(31),CHR\$(71) en CHR\$(177) het cijfer 1 voorstellen.

Het "normale" ASCII Screen alfabet van de APPLE omvat de ASCII code 160 t/m 208. Adres \$32 staat dan op \$FF.

Hoe krijgen we nu een letter in het TEXT-venster. Keer eerst even terug naar Applesoft Basic. Wanneer we de juiste ASCII-waarde naar de juiste geheugenplaats POKEn, dan volgt daaruit een karakterafbeelding op een berekende plaats. Logisch, maar zo op het eerste gezicht wat moeilijk....

```
]10 POKE 1024,193:POKE1025,194:POKE1026,195
]20 POKE 1064,193:POKE1065,194:POKE1066,195
RUN
```

\$400 1024
 \$480 1152
 \$500 1280
 \$580 1408
 \$600 1536
 \$680 1664
 \$700 1792
 \$780 1920
 \$428 1064
 \$4A8 1192
 \$528 1320
 \$5A8 1448
 \$628 1576
 \$6A8 1704
 \$728 1832
 \$7A8 1960
 \$450 1104
 \$4D0 1232
 \$550 1360
 \$5D0 1488
 \$650 1616
 \$6D0 1744
 \$750 1872
 \$7D0 2000

	0	\$00
	1	\$01
	2	\$02
	3	\$03
	4	\$04
	5	\$05
	6	\$06
	7	\$07
	8	\$08
	9	\$09
	10	\$0A
	11	\$0B
	12	\$0C
	13	\$0D
	14	\$0E
	15	\$0F
	16	\$10
	17	\$11
	18	\$12
	19	\$13
	20	\$14
	21	\$15
	22	\$16
	23	\$17
	24	\$18
	25	\$19
	26	\$1A
	27	\$1B
	28	\$1C
	29	\$1D
	30	\$1E
	31	\$1F
	32	\$20
	33	\$21
	34	\$22
	35	\$23
	36	\$24
	37	\$25
	38	\$26
	39	\$27

Map of the Text Screen

Tweemaal het A,B,C. En op twee verschillende regels.
Om inzicht te krijgen in de schermbeschrijving door de
APPLE II maken we een nieuw programma.

```
NEW
HOME
]10 FOR X=1024 TO 2047
20 POKE X,194
30 NEXT X
RUN
```

Een leerzame ervaring. Maar om dit vanuit de monitor te doen, komt er nog iets anders kijken. Wanneer we adres \$400 met de hand de hex-waarde van een "B" geven, ofwel \$C2, en terugkeren naar Basic, zien we geen karakteroutput. Kortom: we zullen in de monitor een apart programma moeten schrijven, dat daarvoor zorgt. Zijn we via een CALL-151 in de monitor (*=prompt), dan kunnen we een aantal machinetaal instructies opgeven, die karakteroutput mogelijk maken. De lokaties \$300 tot \$3EF zijn zeer geschikt voor korte machinetaal programma's. Je zult even moeten wennen aan de typische 6502-instructies. Laten we het eens proberen.

```
*300:20 58 FC    JUMP naar monitorroutine FC58
303:A9 C2        Laadt accumulator met "B"
305:8D 00 04     Store/bewaar "B" op adres $400
308:60          Return naar operating system
```

Om dit programma te laten werken, geef je het monitorcommando voor programmastart: <startadres>G.
Hier: 300G.

In een flits wordt het scherm schoongeveegd (\$FC58, CALL-936 weet je nog), en wordt de B links boven afgebeeld. Het begrip "accumulator" heeft betrekking op de bouw en werking van de 6502-chip. Deze processor voert rekenkundige en logische bewerkingen uit door tussenkomst van de accumulator. Een soort geheugenadres, een register, dat veelvuldig door de 6502 wordt gebruikt. De "accu" kan 1 byte met gegevens (data, of een instructie) bevatten. Omdat dit voor de verwerking van veel gegevens wat weinig is, beschikt de 6502 tevens over een X en een Y register. Deze registers worden vaak als "teller" gebruikt en ze kunnen dan als "index" worden aangewend voor het bijhouden van uitgebreide "loops" etc. De wijze, waarop machinetaal instructies kunnen worden doorgespeeld naar de 6502 (zie je dat we het woord "APPLE" nu opeens kunnen vergeten), wordt op een bijzondere manier benoemd. De meest voorkomende vormen zijn de z.g. impliciete, immediate, absolute en zero page adressering. Een impliciete instructie is de RTS, met Opcode (operatie code) 60. De STA (store in accumulator) instructie, met Opcode 8D, geeft het absolute adres, waarin data moet worden bewaard. LDA (laadt accumulator) wordt onmiddellijk gevolgd door de data; immediate adressering dus. Dit is zeer globaal de werking van de 6502 processor en hangt ten nauwste samen met de werking van de monitor.

Wanneer je direkt na het activeren van de monitor een I tikt, ontstaat een inverse karakteroutput. Een N herstelt deze mode.

adresT trace het programma vanaf adres tot BRK commando
adresS single step door het programma; telkens S return
 ingeven voor volgende stap.

De volgende oefening behelst het oproepen van een monitorroutine, die het toetsenbord leest, een ingebouwde routine, die het teken weergeeft en een routine, die de ASCII-waarde afbeeldt. Het is een oefening, die afkomstig is uit het leuke boek van Don en Kurt Inman "APPLE MACHINE LANGUAGE".

```
*300:A2 00        zet het X-register op 0
:20 58 FC        clear het beeldscherm-HOME
:20 35 FC        JSR naar leesroutine keyboard
:8D 40 03        STA toets op adres $340 (zie lsb/msb!)
:20 ED FD        JSR naar scherm-printroutine COUT
:A9 A0           LDA laadt accu met A0=spatie
:20 ED FD        druk hem ook af
:A9 A0           LDA met nog een A0=spatie
:20 ED FD        vooruit maar - druk hem af
:AD 40 03        LDA met toets vanuit adres $340
:20 DA FD        JSR naar ASCII-print routine
:A9 8D           LDA met waarde voor Return $8D
:20 ED FD        voer de Return uit
:E8              INX ofwel: X=X+1 increment X-register
:E0 1A           CPX compair X, vergelijk X met 26 (dec)
:D0 DD           indien niet gelijk dan 35 stappen terug
                 (FF-DD)
:60              RTS return from subroutine
```

*300G

Hiermee is het mogelijk het hele alfabet in te lezen en de ASCII-waarden ervan te bekijken.

Een bijzonder adres is \$3F8. Hier is plaats voor een JMP-instructie, die verwijzen kan naar een machinetaal routine, die jezelf hebt gespecificeerd. Bijv.

\$3F8:4C D0 03

Door nu CTRL-Y te tikken, dus CTRL-toets tegelijk met de y-toets indrukken, spring je via \$3F8 naar monitorroutine \$3D0, die ervoor zorgt, dat je in Applesoft terugkomt. Gemakkelijk en praktisch.

Nu we toch in Applesoft zijn, kunnen we proberen ons machinetaal programma, dat we net geassembleerd hebben, te gebruiken.

Hoe gaat dat ook alweer ? O ja, met de CALL instructie. Dus CALL768 (return). En uiteindelijk kom je in Basic terug via de RTS.

5. MINI-ASSEMBLER

Indien je een APPLE II in de standaardconfiguratie hebt, dus met Integer Basic in ROM en een Applesoftkaart, dan ben je goed af. Met het Integer Basic komt nml. een Mini-Assembler mee, die je het ongemak van het hand-assembleren kan besparen. Degenen, die een APPLE II PLUS, of EURAPPLE, bezitten, zullen wat extra hard- of software nodig hebben. Een Integer ROM kaart, een Language kaart of een geschikt Integer Software Programma zijn ook voldoende. We gaan ervan uit, dat de mini-assembler in ROM aanwezig is. Het entry-point van de Mini-Assembler is \$F666.

Dus tikken we in de monitor *F666G. De prompt wordt nu een uitroepteken (!). In Basic voldoet een CALL-2458 ook.

Je kunt verder elk monitorcommando uitvoeren, mits het wordt voorafgegaan door een \$-teken.

Ontstaat een tikfout, dan signaleert de Mini-Assembler dit direkt en volgt een foutmelding ('biep') en een caret onder de fout (^).

De Mini-Assembler herinnert zich maar 1 ding: de programmateller. Het is daarom nodig eerst en vooral het adres in te tikken, waar je programma zal moeten beginnen. Tik het in en sluit het adres af met een dubbele punt (:). Hierna mag de instructiecode worden opgegeven in de vorm van een "mnemonic", gevolgd door een spatie. Tenslotte volgt de operand van de instructie. Is dit klaar, dan geef je een tikje op de Returntoets. Hee, da's mooi

```
!300: LDX #$02    (return)
0300- A2 02      LDX #$02
```

De assembler vertaalt de input onmiddellijk tot een geassembleerde regel. Probeer het voorafgaande programma met de Mini-Assembler te programmeren. Omdat gebleken is, dat maar weinigen serieus met de Mini-Assembler werken, zal hier niet uitgebreid bij de functie worden stilgestaan. De adresseringswijze is als voor een 6502 processor: impliciet, immediate, absoluut, zero-page, relatief, indirect geindexeerd, indirect, en indirect genaïndexeerd.

Hier volgt een oefening, die geluiden voortbrengt, wanneer met de game-paddles (spreek uit: geem pedduls) wordt gemanipuleerd.

```

!1D00:LDX # $00
spatie, dan
! JSR FB1E
! STY 1CFF
! INX
! JSR FB1E (lees paddle-monitorroutine)
! LDA C030 (speaker)
! DEC 1CFF (waarde van de paddlestand)
! BNE 1DOC
! LDA C030
! INY
! BNE 1D14
! JMP 1D00

```

Geef nu \$1D00L om het programma te listen en kijk het na. Dat ziet er heel anders uit, dan in de monitor, nietwaar? Met CALL7424 of een 1D00G werkt het programma. Om te stoppen kun je het beste Reset indrukken. Probeer er eens een ander einde aan te maken. BSAVE het programma. Om het op een cassette op te slaan, kun je het beste in de monitor gaan (reset, of \$FF69G).

Tik het begin van het programma in: *1D00 gevolgd door een punt en daarna het eindadres van het programma. Hierna volgt een W voor Write, ofwel: schrijf naar de cassette.

*1D00.1D1CW en wacht met Return tot de cassette opneemt. Druk dan eerst op de Return-toets. Voor het teruglezen van de cassette tik je eerst het begin en eindadres in, maar gevolgd door een R.

*1D00.1D1CR. Start de recorder (play) en tik return.

Step en Trace commando's zijn mogelijk.

Het wijzigen van de 6502 registers gaat als volgt: De registers kunnen slechts in de gedisplayde volgorde worden gewijzigd. Dus A,X,Y,P, en S. Als er een ander dan het A register (de accu) moet worden veranderd, dan moet de inhoud van de voorafgaande registers worden overgenomen. CTRL-E zet de registers open.

Zijn de registers "open", dan wijzig je bijv. P als volgt:

A=3C X=FF Y=00 P=47 S=F4

tik: :3C FF 00 32 (return)

6502 MICROPROCESSOR INSTRUCTIONS

ADC	Add Memory to Accumulator with Carry	LDA	Load Accumulator with Memory
AND	"AND" Memory with Accumulator	LDX	Load Index X with Memory
ASL	Shift Left One Bit (Memory or Accumulator)	LDY	Load Index Y with Memory
BCC	Branch on Carry Clear	LSR	Shift Right one Bit (Memory or Accumulator)
BCS	Branch on Carry Set	NOP	No Operation
BEQ	Branch on Result Zero	ORA	"OR" Memory with Accumulator
BIT	Test Bits in Memory with Accumulator	PHA	Push Accumulator on Stack
BMI	Branch on Result Minus	PHP	Push Processor Status on Stack
BNE	Branch on Result not Zero	PLA	Pull Accumulator from Stack
BPL	Branch on Result Plus	PLP	Pull Processor Status from Stack
BRK	Force Break	ROL	Rotate One Bit Left (Memory or Accumulator)
BVC	Branch on Overflow Clear	ROR	Rotate One Bit Right (Memory or Accumulator)
BVS	Branch on Overflow Set	RTI	Return from Interrupt
CLC	Clear Carry Flag	RTS	Return from Subroutine
CLD	Clear Decimal Mode	SBC	Subtract Memory from Accumulator with Borrow
CLI	Clear Interrupt Disable Bit	SEC	Set Carry Flag
CLV	Clear Overflow Flag	SED	Set Decimal Mode
CMP	Compare Memory and Accumulator	SEI	Set Interrupt Disable Status
CPX	Compare Memory and Index X	STA	Store Accumulator in Memory
CPY	Compare Memory and Index Y	STX	Store Index X in Memory
DEC	Decrement Memory by One	STY	Store Index Y in Memory
DEX	Decrement Index X by One	TAX	Transfer Accumulator to Index X
DEY	Decrement Index Y by One	TAY	Transfer Accumulator to Index Y
EOR	"Exclusive-Or" Memory with Accumulator	TSX	Transfer Stack Pointer to Index X
INC	Increment Memory by One	TXA	Transfer Index X to Accumulator
INX	Increment Index X by One	TXS	Transfer Index X to Stack Pointer
INY	Increment Index Y by One	TYA	Transfer Index Y to Accumulator
JMP	Jump to New Location		
JSR	Jump to New Location Saving Return Address		

THE FOLLOWING NOTATION APPLIES TO THIS SUMMARY:

A	Accumulator
X, Y	Index Registers
M	Memory
C	Borrow
P	Processor Status Register
S	Stack Pointer
✓	Change
—	No Change
+	Add
Λ	Logical AND
-	Subtract
⊕	Logical Exclusive Or
↓	Transfer From Stack
↑	Transfer To Stack
→	Transfer To
←	Transfer To
V	Logical OR
PC	Program Counter
PCH	Program Counter High
PCL	Program Counter Low
OPER	Operand
#	Immediate Addressing Mode

FIGURE 1 ASL-SHIFT LEFT ONE BIT OPERATION

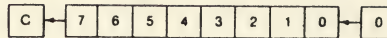


FIGURE 2. ROTATE ONE BIT LEFT (MEMORY OR ACCUMULATOR)

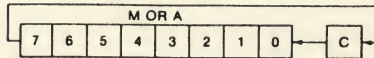
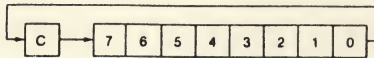


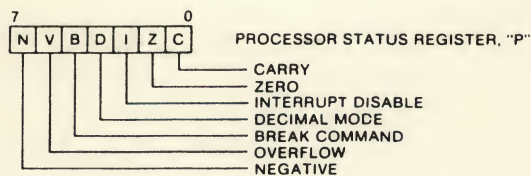
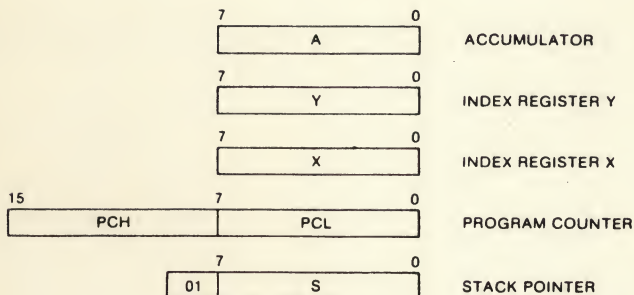
FIGURE 3.



NOTE 1: BIT — TEST BITS

Bit 6 and 7 are transferred to the status register. If the result of $A \wedge M$ is zero then $Z=1$, otherwise $Z=0$.

PROGRAMMING MODEL



Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"P" Status Reg N Z C I O V
ADC Add memory to accumulator with carry	A-M-C → A+C	Immediate Zero Page Zero Page,X Absolute Absolute,X Absolute,Y Absolute,X (Indirect),Y (Indirect),X	ADC #Oper ADC Oper,X ADC Oper,X ADC Oper,X ADC Oper,Y ADC Oper,X ADC (Oper),X ADC (Oper),Y	69 65 75 6D 7D 79 61 71	2 2 2 3 3 3 2 2	V/V/-/-/-V
AND "AND" memory with accumulator	A A M → A	Immediate Zero Page Zero Page,X Absolute Absolute,X Absolute,Y Absolute,X (Indirect),Y (Indirect),X	AND #Oper AND Oper AND Oper,X AND Oper,X AND Oper,X AND Oper,Y AND (Oper),X AND (Oper),Y	29 25 35 2D 3D 39 21 31	2 2 2 3 3 3 2 2	V/V/-/-/-
ASL Shift left one bit (Memory or Accumulator)	(See Figure 1)	Accumulator Zero Page Zero Page,X Absolute Absolute,X	ASL A ASL Oper ASL Oper,X ASL Oper,X ASL Oper,X	0A 06 16 0E 1E	1 2 2 2 3	V/V/V/-/-/-
BCC Branch on carry clear	Branch on C=0	Relative	BCC Oper	90	2	-/-/-/-/-
BCS Branch on carry set	Branch on C=1	Relative	BCS Oper	80	2	-/-/-/-/-
BEQ Branch on result zero	Branch on Z=1	Relative	BEQ Oper	F0	2	-/-/-/-/-
BIT Test bits in memory with accumulator	A A M, M ₇ → N, M ₆ → V	Zero Page Absolute	BIT* Oper BIT* Oper	24 2C	2 3	M ₇ V/-/-/-M ₆
BMI Branch on result minus	Branch on N=1	Relative	BMI Oper	30	2	-/-/-/-/-
BNE Branch on result not zero	Branch on Z=0	Relative	BNE Oper	D0	2	-/-/-/-/-
BPL Branch on result plus	Branch on N=0	Relative	BPL Oper	10	2	-/-/-/-/-
BRK Force Break	Forced Interrupt PC-2 ↑ P ↑	Implied	BRK*	00	1	-/-/-1/-/-
BVC Branch on overflow clear	Branch on V=0	Relative	BVC Oper	50	2	-/-/-/-/-

Note 1 *M₆ and V are reserved to the status register if the result of A V M is
Note 2 *A BRK command cannot be masked by setting V

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"P" Status Reg N Z C I O V
BVS Branch on overflow set	Branch on V=1	Relative	BVS Oper	70	2	-/-/-/-/-
CLC Clear carry flag	0 → C	Implied	CLC	18	1	-/-/-0/-/-
CLD Clear decimal mode	0 → D	Implied	CLD	D8	1	-0/-/-/-/-
CLI	0 → I	Implied	CLI	58	1	-/-0/-/-/-
CLV Clear overflow flag	0 → V	Implied	CLV	88	1	0/-/-/-/-/-
CMP Compare memory and accumulator	A → M	Immediate Zero Page Zero Page,X Absolute Absolute,X Absolute,Y Absolute,X (Indirect),Y (Indirect),X	CMP #Oper CMP Oper CMP Oper,X CMP Oper,X CMP Oper,X CMP Oper,Y CMP (Oper),X CMP (Oper),Y	C9 C5 D5 CD DD D9 C1 D1	2 2 2 3 3 3 2 2	V/V/V/-/-/-
CPX Compare memory and index X	X → M	Immediate Zero Page Absolute	CPX #Oper CPX Oper CPX Oper	E0 E4 EC	2 2 3	V/V/V/-/-/-
CPY Compare memory and index Y	Y → M	Immediate Zero Page Absolute	CPY #Oper CPY Oper CPY Oper	C0 C4 CC	2 2 3	V/V/V/-/-/-
DEC Decrement memory by one	M → 1 → M	Zero Page Zero Page,X Absolute Absolute,X	DEC Oper DEC Oper,X DEC Oper DEC Oper,X	C8 D8 CE DE	2 2 3 3	V/V/-/-/-/-
DEX Decrement index X by one	X → 1 → X	Implied	DEX	CA	1	V/V/-/-/-/-
DEY Decrement index Y by one	Y → 1 → Y	Implied	DEY	88	1	V/V/-/-/-/-

INSTRUCTION CODES

INSTRUCTION CODES

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"r" Status Flag N Z C I O V
EOR Exclusive-Or memory with accumulator	A V M \rightarrow A	Immediate Zero Page Zero Page,X Absolute Absolute,X Absolute,Y EOR (Oper,X) (Indirect,X) EOR (Oper,Y) EOR (Oper,X) EOR (Oper,Y)	EOR #Oper EOR Oper,X EOR Oper,X EOR Oper,X EOR Oper,X EOR Oper,X EOR (Oper,X) EOR (Oper,X) EOR (Oper,X) EOR (Oper,X)	49 45 55 4D 5D 50 59 41 51	2 2 2 3 3 3 2 2 2	✓✓-----
INC Increment memory by one	M + 1 \rightarrow M	Zero Page Zero Page,X Absolute Absolute,X	INC Oper INC Oper,X INC Oper INC Oper,X	E6 F6 EE FE	2 2 3 3	✓✓-----
INX Increment index X by one	X + 1 \rightarrow X	Implied	INX	EB	1	✓✓-----
INY Increment index Y by one	Y + 1 \rightarrow Y	Implied	INY	CB	1	✓✓-----
JMP Jump to new location	(PC-1) \rightarrow PC (PC-2) \rightarrow PC	Absolute Indirect	JMP Oper JMP (Oper)	4C 6C	3 3	-----
JSR Jump to new location saving return address	PC-2 \downarrow (PC-1) \rightarrow PC (PC-2) \rightarrow PC	Absolute	JSR Oper	20	3	-----
LDA Load accumulator with memory	M \rightarrow A	Immediate Zero Page Zero Page,X Absolute Absolute,X Absolute,Y Absolute,X (Indirect,X) LDA (Oper,X) LDA (Oper,Y)	LDA #Oper LDA Oper LDA Oper,X LDA Oper,X LDA Oper,X LDA Oper,X LDA Oper,X LDA (Oper,X) LDA (Oper,X) LDA (Oper,X)	A9 A5 B5 AD BD B9 A1 B1	2 2 2 3 3 3 2 2 2	✓✓-----
LDX Load index X with memory	M \rightarrow X	Immediate Zero Page Zero Page,X Absolute Absolute,X	LDX #Oper LDX Oper LDX Oper,X LDX Oper,X LDX Oper,X	A2 A6 B6 AE BE	2 2 2 3 3	✓✓-----
LDY Load index Y with memory	M \rightarrow Y	Immediate Zero Page Zero Page,X Absolute Absolute,X	LDY #Oper LDY Oper LDY Oper,X LDY Oper,X LDY Oper,X	A0 A4 B4 AC BC	2 2 2 3 3	✓✓-----

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX OP Code	No. Bytes	"r" Status Flag N Z C I O V
LSR Shift right one bit (memory or accumulator)	(See Figure 1)	Accumulator Zero Page Zero Page,X Absolute Absolute,X	LSR A LSR Oper LSR Oper,X LSR Oper,X LSR Oper,X	4A 46 56 4E 5E	1 2 2 3 3	0✓✓-----
NOP No operation	No Operation	Implied	NOP	EA	1	-----
ORA Or memory with accumulator	A V M \rightarrow A	Immediate Zero Page Zero Page,X Absolute Absolute,X Absolute,Y Absolute,X (Indirect,X) ORA (Oper,X) ORA (Oper,Y)	ORA #Oper ORA Oper ORA Oper,X ORA Oper,X ORA Oper,X ORA Oper,X ORA Oper,X ORA (Oper,X) ORA (Oper,X) ORA (Oper,X)	09 05 15 00 10 19 01 11	2 2 2 3 3 3 2 2 2	✓✓-----
PHA Push accumulator on stack	A \uparrow	Implied	PHA	48	1	-----
PHP Push processor status on stack	P \uparrow	Implied	PHP	08	1	-----
PLA Pull accumulator from stack	A \uparrow	Implied	PLA	68	1	✓✓-----
PLP Pull processor status from stack	P \uparrow	Implied	PLP	28	1	From Stack
ROL Rotate one bit left (memory or accumulator)	(See Figure 2)	Accumulator Zero Page Zero Page,X Absolute Absolute,X	ROL A ROL Oper ROL Oper,X ROL Oper,X ROL Oper,X	2A 26 36 2E 3E	1 2 2 3 3	✓✓✓-----
ROR Rotate one bit right (memory or accumulator)	(See Figure 3)	Accumulator Zero Page Zero Page,X Absolute Absolute,X	ROR A ROR Oper ROR Oper,X ROR Oper,X ROR Oper,X	6A 66 76 BE 7E	1 2 2 3 3	✓✓✓-----

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX Op Code	No. Bytes	"P" Status Reg. N Z C I O V
RTI Return from interrupt	P ← PC ↑	Implied	RTI	40	1	From Stack
RTS Return from subroutine	PC ↑, PC-1 → PC	Implied	RTS		1	-----
SBC Subtract memory from accumulator with borrow	A ← M - C → A	Immediate Zero Page Zero Page X Absolute Absolute X Absolute Y SBC Oper. Y SBC (Oper. X) (Indirect. Y)	SBC #Oper SBC Oper SBC Oper. X SBC Oper SBC Oper. X SBC Oper. Y SBC (Oper. X) SBC (Oper. Y)	E9 E5 E5 ED ED F9 F9 F1	2 2 2 3 3 3 2 2	✓✓✓---\
SEC Set carry flag	1 → C	Implied	SEC	38	1	---1---
SED Set decimal mode	1 → D	Implied	SED	F8	1	----1-
SEI Set interrupt disable status	1 → I	Implied	SEI	78	1	---1---
STA Store accumulator in memory	A → M	Zero Page Zero Page X Absolute Absolute X Absolute Y (Indirect. X) (Indirect. Y)	STA Oper STA Oper. X STA Oper STA Oper. X STA Oper. Y STA (Oper. X) STA (Oper. Y)	85 95 80 90 99 81 91	2 2 3 3 3 2 2	-----
STX Store index X in memory	X → M	Zero Page Zero Page Y Absolute	STX Oper STX Oper. Y STX Oper	86 98 8E	2 2 3	-----
STY Store index Y in memory	Y → M	Zero Page Zero Page X Absolute	STY Oper STY Oper. X STY Oper	B4 94 8C	2 2 3	-----
TAX Transfer accumulator to index X	A → X	Implied	TAX	AA	1	✓✓-----
TAY Transfer accumulator to index Y	A → Y	Implied	TAY	A8	1	✓✓-----
TSX Transfer stack pointer to index X	S → X	Implied	TSX	BA	1	✓✓-----

Name Description	Operation	Addressing Mode	Assembly Language Form	HEX Op Code	No. Bytes	"P" Status Reg. N Z C I O V
TXA Transfer index X to accumulator	X → A	Implied	TXA	8A	1	✓✓-----
TXS Transfer index X to stack pointer	X → S	Implied	TXS	9A	1	-----
TYA Transfer index Y to accumulator	Y → A	Implied	TYA	98	1	✓✓-----

INSTRUCTION CODES

HEX OPERATION CODES

00 — BRK	2F — NOP	5E — LSR — Absolute, X	8D — STA — Absolute	B4 — LDY — Zero Page, X	DB — NOP
01 — ORA — (indirect, X)	30 — BMI	5F — NOP	8E — STX — Absolute	B5 — LDA — Zero Page, X	DC — NOP
02 — NOP	31 — AND — (indirect, Y)	60 — RTS	8F — NOP	B6 — LDX — Zero Page, Y	DD — CMP — Absolute, X
03 — NOP	32 — NOP	61 — ADC — (indirect, X)	90 — BCC	B7 — NOP	DE — DEC — Absolute, X
04 — NOP	33 — NOP	62 — NOP	91 — STA — (indirect, Y)	B8 — CLV	DF — NOP
05 — ORA — Zero Page	34 — NOP	63 — NOP	92 — NOP	B9 — LDA — Absolute, Y	E0 — CPX — Immediate
06 — ASL — Zero Page	35 — AND — Zero Page, X	64 — NOP	93 — NOP	BA — TSX	E1 — SBC — (indirect, X)
07 — NOP	36 — ROL — Zero Page, X	65 — ADC — Zero Page	94 — STA — Zero Page, X	BB — NOP	E2 — NOP
08 — PHP	37 — NOP	66 — ROR — Zero Page	95 — STA — Zero Page, X	BC — LDY — Absolute, X	E3 — NOP
09 — ASL — Immediate	38 — SEC	67 — NOP	96 — STX — Zero Page, Y	BD — LDA — Absolute, X	E4 — CPX — Zero Page
0A — ORA — Accumulator	39 — AND — Absolute, Y	68 — PLA	97 — NOP	BE — LDX — Absolute, Y	E5 — SBC — Zero Page
0B — NOP	3A — NOP	69 — ADC — Immediate	98 — TYA	BF — NOP	E6 — INC — Zero Page
0C — NOP	3B — NOP	6A — ROR — Accumulator	99 — STA — Absolute, Y	C0 — CPY — Immediate	E7 — NOP
0D — ORA — Absolute	3C — NOP	6B — NOP	9A — TXS	C1 — CMP — (indirect, X)	E8 — INX
0E — ASL — Absolute	3D — AND — Absolute, X	6C — JMP — Indirect	9B — NOP	C2 — NOP	E9 — SBC — Immediate
0F — NOP	3E — ROL — Absolute, X	6D — ADC — Absolute	9C — NOP	C3 — NOP	EA — NOP
10 — BPL	3F — NOP	6E — ROR — Absolute	9D — STA — Absolute, X	C4 — CPY — Zero Page	EB — NOP
11 — ORA — (indirect, Y)	40 — RPL	6F — NOP	9E — NOP	C5 — CMP — Zero Page	EC — CPX — Absolute
12 — NOP	41 — EOR — (indirect, X)	70 — BVS	9F — NOP	C6 — DEC — Zero Page	ED — SBC — Absolute
13 — NOP	42 — NOP	71 — ADC — (indirect, Y)	A0 — LDY — Immediate	C7 — NOP	EE — INC — Absolute
14 — NOP	43 — NOP	72 — NOP	A1 — LDA — (indirect, X)	C8 — INY	EF — NOP
15 — ORA — Zero Page, X	44 — NOP	73 — NOP	A2 — LDX — Immediate	C9 — CMP — Immediate	F0 — BEQ
16 — ASL — Zero Page, X	45 — EOR — Zero Page	74 — NOP	A3 — NOP	CA — DEX	F1 — SBC — (indirect, Y)
17 — NOP	46 — LSR — Zero Page, X	75 — ADC — Zero Page, X	A4 — LDY — Immediate	CB — NOP	F2 — NOP
18 — CLC	47 — NOP	76 — ROR — Zero Page, X	A5 — LDA — Zero Page	CC — CPY — Absolute	F3 — NOP
19 — ORA — Absolute, Y	48 — PHA	77 — NOP	A6 — LDX — Zero Page	CD — CMP — Absolute	F4 — NOP
1A — NOP	49 — EOR — Immediate	78 — SEI	A7 — NOP	CE — DEC — Absolute	F5 — SBC — Zero Page, X
1B — NOP	4A — LSR — Accumulator	79 — ADC — Absolute, Y	A8 — TAY	CF — NOP	F6 — INC — Zero Page, X
1C — NOP	4B — NOP	7A — NOP	A9 — LDA — Immediate	D0 — BNE	F7 — NOP
1D — ORA — Absolute, X	4C — JMP — Absolute	7B — NOP	AA — TAX	D1 — CMP — (indirect, Y)	F8 — SED
1E — ASL — Absolute, X	4D — EOR — Absolute	7C — NOP	AB — NOP	D2 — NOP	F9 — SBC — Absolute, Y
1F — NOP	4E — LSR — Absolute	7D — ADC — Absolute, X NOP	AC — LDY — Absolute	D3 — NOP	FA — NOP
20 — JSR	4F — NOP	7E — ROR — Absolute, X NOP	AD — Absolute	D4 — NOP	FB — NOP
21 — AND — (indirect, X)	50 — BVC	7F — NOP	AE — LDX — Absolute	D5 — CMP — Zero Page, X	FC — NOP
22 — NOP	51 — EOR (indirect, Y)	80 — NOP	AF — NOP	D6 — DEC — Zero Page, X	FD — SBC — Absolute, X
23 — NOP	52 — NOP	81 — STA — (indirect, X)	B0 — BCS	D7 — NOP	FE — INC — Absolute, X
24 — BIT — Zero Page	53 — NOP	82 — NOP	B1 — LDA — (indirect, Y)	D8 — CLD	FF — NOP
25 — AND — Zero Page	54 — NOP	83 — NOP	B2 — NOP	D9 — CMP — Absolute, Y	
26 — ROL — Zero Page	55 — EOR — Zero Page, X	84 — STY — Zero Page	B3 — NOP		
27 — NOP	56 — LSR — Zero Page, X	85 — STA — Zero Page			
28 — PLP	57 — NOP	86 — STX — Zero Page			
29 — AND — Immediate	58 — CLI	87 — NOP			
2A — ROL — Accumulator	59 — EOR — Absolute, Y	88 — DEY			
2B — NOP	5A — NOP	89 — NOP			
2C — BIT — Absolute	5B — NOP	8A — TXA			
2D — AND — Absolute	5C — NOP	8B — NOP			
2E — ROL — Absolute	5D — EOR — Absolute, X	8C — STY — Absolute			


```

*****
*
*      APPLE-II
*      MINI-ASSEMBLER
*
*      COPYRIGHT 1977 BY
*      APPLE COMPUTER INC.
*
*      ALL RIGHTS RESERVED
*
*      S. WOZNIAK
*      A. BAUM
*
*****

```

```

*****
TITLE "APPLE-II MINI-ASSEMBLER"
*****

```

```

FORAAT      EPZ  S2F
LENGTH      EPZ  S2F
MODE        EPZ  S31
PROEPT      EPZ  S33
YSAV        EPZ  S34
L           EPZ  S35
PCL         EPZ  S3A
PCH         EPZ  S3B
A1H         EPZ  S3C
A2L         EPZ  S3F
A2H         EPZ  S3F
A4L         EPZ  S42
A4H         EPZ  S43
FMT         EPZ  S44
IN          EQU  S200
INSDS2      EQU  SF6FE
INSTO3F     EQU  SF6D0
PR3L2       EQU  SF94A
PCADJ       EQU  SF953
CHAR1       EQU  SF9B4
CHAR2       EQU  SF9FA
MNEB1L     EQU  SF9C0
MNEAP      EQU  SFA00
CURSUP      EQU  SFC1A
GETLN2      EQU  SFD67
COUT        EQU  SFD6D
BL1         EQU  SFE00
ALPCLP      EQU  SFE76
ELL         EQU  SFF3A
GETNUM      EQU  SFFA7
TOSUP       EQU  SFFFF
ZMODE       EQU  SFFC7
CHPTOL      EQU  SFFCC
ORG         ORG  SF500
REL         SRC  #S21
LSP         LSP  A
LJL         LJL  #R3
LDY         LDY  A2:
LDY         LDY  A2L
BNE         BNE  REL2
DEFY        DEFY
REL2        DEX
TXA         TXA
CLC         CLC
SBC         SBC  PCL
STA         STA  A2L
BPL         BPL  REL3
INY         INY
REL3        TYA

```

```

F500: E9 61
F502: 4A
F503: D0 14
F505: A4 3F
F507: A6 3E
F509: D0 01
F50B: 88
F50C: CA
F50D: 8A
F50E: 18
F50F: E5 3A
F511: 85 3E
F513: 10 01
F515: C8
F516: 98

```

```

IS FMT COMPATIBLE
WITH RELATIVE MODE?
NO.

```

```

DOUBLE DECREMENT

```

```

FORM ADDR-PC-2

```

F517:	E5 3B		SBC	PCH	
F519:	D0 6B	ERR3	RNE	ERR	ERROR IF >1-BYTE BRANCH
F51B:	A4 2F	FINDOP	LDY	LENGTH	
F51D:	B9 3D 00	FNDOP2	LDA	ALH,Y	MOVE INST TO (PC)
F520:	91 3A		STA	(PCL),Y	
F522:	88		DEY		
F523:	10 F8		BPL	FNDOP2	
F525:	20 1A FC		JSR	CURSOP	
F528:	20 1A FC		JSR	CURSOP	RESTORE CURSOR
F52B:	20 D0 F8		JSR	INSTDSP	TYPE FORMATTED LINE
F52E:	20 53 F9		JSR	PCADJ	UPDATE PC
F531:	84 3B		STY	PCH	
F533:	85 3A		STA	PCL	
F535:	4C 95 F5		JMP	NXTLINE	GET NEXT LINE
F538:	20 BE FF	FAKEMON3	JSR	TOSUB	GO TO DELIM HANDLER
F53B:	A4 34		LDY	YSAV	RESTORE Y-INDEX
F53D:	20 A7 FF	FAKEMON	JSR	GETNUM	READ PARAM
F540:	84 34		STY	YSAV	SAVE Y-INDEX
F542:	A0 17		LDY	#S17	INIT DELIMITER INDEX
F544:	88	FAKEMON2	DEY		CHECK NEXT DELIM
F545:	30 4B		BMI	PESETZ	ERR IF UNRECOGNIZED DELIM
F547:	D9 CC FF		CMP	CHRTBL,Y	COMPARE WITH DELIM TABLE
F54A:	D0 F8		RNE	FAKEMON2	NO MATCH
F54C:	C0 15		CPY	#S15	MATCH, IS IT CR?
F54E:	D0 E8		BNE	FAKEMON3	NO, HANDLE IT IN MONITOR
F550:	A5 31		LDA	MODE	
F552:	A0 00		LDY	#S0	
F554:	C6 34		DEC	YSAV	
F556:	20 00 FE		JSR	BL1	HANDLE CR OUTSIDE MONITOR
F559:	4C 95 F5		JMP	NXTLINE	
F55C:	A5 3D	TRYNEXT	LDA	ALH	GET TRIAL OPCODE
F55E:	20 8E F8		JSR	INSDS2	GET FMT+LENGTH FOR OPCODE
F561:	AA		TAX		
F562:	BD 00 FA		LDA	MNEMR,X	GET LOWER MNEMONIC BYTE
F565:	C5 42		CMP	A4L	MATCH?
F567:	D0 13		RNE	NEXTOP	NO, TRY NEXT OPCODE
F569:	BD C0 F9		LDA	MNEML,X	GET UPPER MNEMONIC BYTE
F56C:	C5 43		CMP	A4H	MATCH?
F56E:	D0 0C		RNE	NEXTOP	NO, TRY NEXT OPCODE.
F570:	A5 44		LDA	FMT	
F572:	A4 2E		LDY	FORMAT	GET TRIAL FORMAT
F574:	C0 9D		CPY	#S9D	TRIAL FORMAT RELATIVE?
F576:	F0 88		BEQ	REL	YES.
F578:	C5 2E	NREL	CMP	FORMAT	SAME FORMAT?
F57A:	F0 9F		BEQ	FINDOP	YES.
F57C:	C6 3D	NEXTOP	DEC	ALH	NO, TRY NEXT OPCODE
F57E:	D0 DC		BNE	TRYNEXT	
F580:	E6 44		INC	FMT	NO MORE, TRY WITH LEN=2
F582:	C6 35		DEC	L	WAS L=2 ALREADY?
F584:	F0 D6		BEQ	TRYNEXT	NO.
F586:	A4 34	ERR	LDY	YSAV	YES, UNRECOGNIZED INST.
F588:	98	ERR2	TYA		
F589:	AA		TAX		
F58A:	20 4A F9		JSR	PRBL2	PRINT ~ UNDER LAST READ
F58D:	A9 DE		LDA	#SDE	CHAR TO INDICATE ERROR
F58F:	20 ED FD		JSR	COUT	POSITION.
F592:	20 3A FF	RESETZ	JSR	BELL	
F595:	A9 A1	NXTLINE	LDA	#SA1	'!'
F597:	85 33		STA	PROMPT	INITIALIZE PROMPT
F599:	20 67 FD		JSR	GETLN2	GET LINE.
F59C:	20 C7 FF		JSR	ZMODE	INIT SCREEN STUFF
F59F:	AD 00 02		LDA	IN	GET CHAR
F5A2:	C9 A0		CMP	#SA0	ASCII BLANK?
F5A4:	F0 13		BEQ	SPACE	YES
F5A6:	C8		INY		
F5A7:	C9 A4		CMP	#SA4	ASCII '\$' IN COL 1?
F5A9:	F0 92		BEQ	FAKEMON	YES, SIMULATE MONITOR
F5AB:	88		DEY		NO, BACKUP A CHAR
F5AC:	20 A7 FF		JSR	GETNUM	GET A NUMMER
F5AF:	C9 93		CMP	#S93	':' TERMINATOR?
F5B1:	D0 D5	ERR4	BNI	ERR2	NO, ERR.
F5B3:	8A		TXA		
F5B4:	F0 D2		BEQ	ERR2	NO ADR PRECEDING COLON.
F5B6:	20 78 FE		JSR	ALPCLF	MOVE ADR TO PCL, PCH.
F5B9:	A9 03	SPACE	LDA	#S3	COUNT OF CHARS IN MNEMONIC
F5BB:	85 3D		STA	ALH	
F5BD:	20 34 F6	NXTMN	JSR	GETNSP	CET FIRST MNEM CHAR.
F5C0:	0A	NXTM	ASL	A	
F5C1:	E9 BE		SEC	#SBE	SUBTRACT OFFSET
F5C3:	C9 C2		CMP	#SC2	LEGAL CHAR?
F5C5:	90 C1		BCC	ERR2	NO.
F5C7:	0A		ASL	A	COMPRESS-LEFT JUSTIFY
F5C8:	0A		ASL	A	
F5C9:	A2 04		LDX	#S4	
F5CB:	0A	NXTM2	ASL	A	DO 5 TRIPLE WORD SHIFTS

F5CC: 26 42		ROL A4L	
F5CE: 26 43		ROL A4H	
F5D0: CA		DEX	
F5D1: 10 F8		BPL NXTM2	
F5D3: C6 3D		DEC A1H	DONF WITH 3 CHARS?
F5D5: F0 F4		BFO NXTM2	YES, BUT DO 1 MORE SHIFT
F5D7: 10 E4		BPL NXTMN	NO
F5D9: A2 05	FORM1	LDX #55	5 CHARS IN ADDR MODE
F5DB: 20 34 F6	FORM2	JSR GETNSP	GET FIRST CHAR OF ADDR
F5DE: 84 34		STY YSAV	
F5E0: DD 34 F9		CMP CHAR1,X	FIRST CHAR MATCH PATTERN?
F5E3: D0 13		BNE FORM3	NO
F5E5: 20 34 F6		JSR GETNSP	YES, GET SECOND CHAR
F5E8: DD 34 F9		CMP CHAR2,X	MATCHES SECOND HALF?
F5EE: F0 0D		BFO FORM5	YES
F5ED: DD 34 F9		LDA CHAR2,X	NO, IS SECOND HALF ZERO?
F5F0: F0 07		BEO FORM4	YES.
F5F2: C9 A4		CMP #5A4	NO, SECOND HALF OPTIONAL?
F5F4: F0 03		BEO FORM4	YES.
F5F6: A4 34		LDY YSAV	
F5F8: 18	FORM3	CLC	CLEAR BIT-NO MATCH
F5F9: 88	FORM4	DEY	BACK UP 1 CHAR
F5FA: 26 44	FORM5	ROL FMT	FORM FORMAT BYTE
F5FC: E0 03		CFX #53	TIME TO CHECK FOR ADDR.
F5FE: D0 0D		BNE FORM7	NO
F600: 20 A7 FF		JSR GETNUM	YES
F603: A5 3F		LDA A2H	
F605: F0 01		BEO FORM6	HIGH-ORDER BYTE ZERO
F607: E9		INX	NO, INCR FOR 2-BYTE
F608: 86 35	FORM6	STX L	STORE LENGTH
F60A: A2 03		LDX #53	RELOAD FORMAT INDEX
F60C: 88		DEY	BACKUP A CHAR
F60D: 86 3D	FORM7	STX A1H	SAVE INDEX
F60F: CA		DEX	DONE WITH FORMAT CHECK?
F610: 10 C9		BPL FORM2	NO.
F612: A5 44		LDA FMT	YES, PUT LENGTH
F614: 0A		ASL A	IN LOW BITS
F615: 0A		ASL A	
F616: 05 35		ORA L	
F618: C9 20		CMP #520	
F61A: 80 06		RCS FORM8	ADD '\$' IF NONZERO LENGTH
F61C: A6 35		LDX L	AND DON'T ALREADY HAVE IT
F61E: F0 02		BEO FORM8	
F620: 09 80		ORA #560	
F622: 85 44	FORM8	STA FMT	
F624: 84 34		STY YSAV	
F626: 89 00 02		LDA IN,Y	GET NEXT NONBLANK
F629: C9 8B		CMP #5B3	' ' START OF COMMENT?
F62B: F0 04		BEO FORM9	YES
F62D: C9 8D		CMP #58D	CARRIAGE RETURN?
F62F: D0 80		BNE ERR4	NO, ERR.
F631: 4C 5C F5	FORM9	JMP TRYNEXT	
F634: 89 00 02	GETNSP	LDA IN,Y	
F637: C8		INY	
F638: C9 A0		CMP #5A0	GET NEXT NON BLANK CHAR
F63A: F0 F8		BEO GETNSP	
F63C: 60		RTS	
		ORG \$F666	
F666: 4C 92 F5	MINASM	JMP RESETZ	


```

*****
*
*      APPLE II
*      SYSTEM MONITOR
*
*      COPYRIGHT 1977 BY
*      APPLE COMPUTER, INC.
*
*      ALL RIGHTS RESERVED
*
*      S. WOZNIAK
*      A. BAUM
*
*****

```

```

*****
*      TITLE      "APPLE II SYSTEM MONITOR"
*
*      LOC0      EPZ $00
*      LOC1      EPZ $01
*      WNDLFT     EPZ $20
*      WNDWDTH    EPZ $21
*      WNDTOP     EPZ $22
*      WNDBTM     EPZ $23
*      CH         EPZ $24
*      CV         EPZ $25
*      GBASL      EPZ $26
*      GBASH      EPZ $27
*      BASL       EPZ $28
*      BASH       EPZ $29
*      BAS2L      EPZ $2A
*      BAS2H      EPZ $2B
*      H2         EPZ $2C
*      LMNEM      EPZ $2C
*      RTNL       EPZ $2C
*      V2         EPZ $2D
*      RMNEM      EPZ $2D
*      RTNH       EPZ $2D
*      MASK       EPZ $2F
*      CHKSUM     EPZ $2F
*      FORMAT     EPZ $2E
*      LASTIN     EPZ $2F
*      LENGTH     EPZ $2F
*      SIGN       EPZ $2F
*      COLOR      EPZ $30
*      MODE       EPZ $31
*      INVFLG     EPZ $32
*      PROMPT     EPZ $33
*      YSAV       EPZ $34
*      YSAV1      EPZ $35
*      CSWL       EPZ $36
*      CSWH       EPZ $37
*      KSWL       EPZ $38
*      KSWH       EPZ $39
*      PCL        EPZ $3A
*      PCH        EPZ $3B
*      XQT        EPZ $3C
*      A1L        EPZ $3C
*      A1H        EPZ $3D
*      A2L        EPZ $3E
*      A2H        EPZ $3F
*      A3L        EPZ $40
*      A3H        EPZ $41
*      A4L        EPZ $42
*      A4H        EPZ $43
*      A5L        EPZ $44
*      A5H        EPZ $45

```

	ACC	EPZ	\$45	
	XIFG	EPZ	\$46	
	YIFG	EPZ	\$47	
	STATUS	EPZ	\$48	
	SPNT	EPZ	\$49	
	RDDL	EPZ	\$4F	
	RNDH	EPZ	\$4F	
	ACL	EPZ	\$50	
	ACH	EPZ	\$51	
	XINDL	EPZ	\$52	
	XTDPH	EPZ	\$53	
	AUXL	EPZ	\$54	
	AUXH	EPZ	\$55	
	PICK	EPZ	\$95	
	IN	EOU	\$0200	
	USHADD	EOU	\$03FE	
	PHI	EOU	\$03FE	
	IFOLOC	EOU	\$03FF	
	ICADR	EOU	\$C000	
	KAD	EOU	\$C000	
	KROSTRP	EOU	\$C010	
	TAPFOUT	EOU	\$C020	
	SPKR	EOU	\$C030	
	TXTCLE	EOU	\$C050	
	TATSET	EOU	\$C051	
	MIXCLF	EOU	\$C052	
	MIXSET	EOU	\$C053	
	LOWSCR	EOU	\$C054	
	HISCR	EOU	\$C055	
	LORES	EOU	\$C056	
	HIRES	EOU	\$C057	
	TAPEIN	EOU	\$C060	
	PADDIO	EOU	\$C064	
	PTRIG	EOU	\$C070	
	BASIC	EOU	\$E000	
	BASIC2	EOU	\$E003	
		ORG	\$F800	POM START ADDRESS
F800:	4A	LSR	A	Y-COORD/2
F801:	08	PHP		SAVE LSB IN CARRY
F802:	20 47 F6	JSR	GBASCALC	CALC BASE ADR IN GBASL,H
F805:	28	PLP		RESTORE LSR FROM CARRY
F806:	A9 0F	LDA	#S0F	MASK S0F IF EVEN
F808:	90 02	BCC	RTMASK	
F80A:	69 E0	ADC	#SE0	MASK \$F0 IF ODD
F80C:	85 2E	STA	MASK	
F80E:	B1 26	LDA	(GBASL),Y	DATA
F810:	45 30	EOP	COLOR	XOR COLOP
F812:	25 2E	AND	MASK	AND MASK
F814:	51 26	EOR	(GBASL),Y	XOR DATA
F816:	91 26	STA	(GBASL),Y	TO DATA
F818:	60	RTS		
F819:	20 00 F8	JSR	PLOT	PLOT SQUARE
F81C:	C4 2C	CPY	H2	DONE?
F81E:	B0 11	RCS	RTS1	YES, RETURN
F820:	C8	INY		NO, INCR INDEX (X-COORD)
F821:	20 0E F6	JSR	PLOT1	PLOT NEXT SQUARE
F824:	90 F6	BCC	HLINE1	ALWAYS TAKEN
F826:	69 01	ADC	#S01	NEXT Y-COORD
F828:	48	PHA		SAVE ON STACK
F829:	20 00 F8	JSR	PLOT	PLOT SQUARE
F82C:	68	PLA		
F82D:	C5 2D	CMP	V2	DONE?
F82F:	90 F5	BCC	VLINEZ	NO, LOOP.
F831:	60	RTS		
F832:	A0 2F	LDY	#S2F	MAX Y, FULL SCRIN CLR
F834:	D0 02	PNE	CLRSC2	ALWAYS TAKEN
F836:	A0 27	LDY	#S27	MAX Y, TOP SCRIN CLR
F838:	84 2D	STY	V2	STORE AS BOTTOM COORD
		FOR	VLINE CALLS	
F83A:	A0 27	LDY	#S27	RIGHTMOST X-COORD (COLUMN)
F83C:	A9 00	LDA	#S0	TOP COORD FOR VLINE CALLS
F83E:	85 30	STA	COLOR	CLEAR COLOR (BLACK)
F840:	20 28 F6	JSR	VLINE	FROM VLINE
F843:	88	DEY		NEXT LEFTMOST X-COORD
F844:	10 F6	AVL	CLRSC3	LOOP UNTIL DONE.
F846:	60	RTS		
F847:	48	GBASCALC	PHA	FOR INPUT 0000FFGH
F848:	4A	LSR	A	
F849:	29 03	AND	#S03	
F84B:	09 04	ORA	#S04	GENERATE GBASH=000001FG
F84D:	85 27	STA	GBASH	
F84F:	68	PLA		AND GBASL=0000F000
F850:	29 18	AND	#S18	
F852:	90 02	SCC	GBASCALC	
F854:	69 7F	ADC	#S7F	
F856:	85 26	GBASCALC	STA	GBASL

F858:	0A		ASL	A	
F859:	0A		ASL	A	
F85A:	05 26		ORA	GRASL	
F85C:	85 26		STA	GRASL	
F85E:	60		RTS		
F85F:	A5 30	NXICOL	LDA	COLOR	INCREMENT COLOR BY 3
F861:	13		CLC		
F862:	69 03		ADC	#S03	
F864:	29 0F	SETCOL	AND	#S0F	SETS COLOR=17*A MOD 16
F866:	85 30		STA	COLOR	
F868:	0A		ASL	A	BOTH HALF BYTES OF COLOR EQUAL
F869:	0A		ASL	A	
F86A:	0A		ASL	A	
F86B:	0A		ASL	A	
F86C:	05 30		ORA	COLOR	
F86E:	85 30		STA	COLOR	
F870:	60		RTS		
F871:	4A	SCFN	LSR	A	READ SCREEN Y-COORD/2
F872:	06		PHP		SAVE LSB (CARRY)
F873:	20 47 F8		JSR	GRASCALC	CALC BASE ADDRESS
F876:	B1 26		LDA	(GRASL),Y	GET BYTE
F878:	28		PLP		RESTORE LSP FROM CARRY
F879:	90 04	SCRN2	RCC	RTESKZ	IF EVEN, USE LO H
F87B:	4A		LSR	A	
F87C:	4A		LSR	A	
F87D:	4A		LSR	A	SHIFT HIGH HALF BYTE DOWN
F87E:	4A		LSR	A	
F87F:	29 0F	RTMSKZ	AND	#S0F	MASK 4-BITS
F881:	60		RTS		
F882:	A6 3A	INSDS1	LDX	PCL	PRINT PCL,H
F884:	A4 38		LDY	PCY	
F886:	20 96 FD		JSP	PRYX2	
F889:	20 40 F9		JSR	PRBLK	FOLLOWED BY A BLANK
F88C:	A1 3A		LDA	(PCL,X)	GET OP CODE
F88E:	A8	INSDS2	TAY		
F88F:	4A		LSR	A	EVEN/ODD TEST
F890:	90 09		RCC	IEVEN	
F892:	6A		ROP	A	BIT 1 TEST
F893:	80 10		BCC	ERR	XXXXXX11 INVALID OP
F895:	C9 12		CMP	#S02	
F897:	F0 0C		FEO	ERR	OPCODE S09 INVALID
F899:	29 87		AND	#S07	MASK BITS
F89B:	4A	IEVEN	LSR	A	LSR INTO CARRY FOR I/O TEST
F89C:	AA		TAX		
F89D:	BC 62 F9		LDA	FM1,X	GET FORMAT INDEX BYTE
F8A0:	20 79 F8		JSP	SCRN2	P/L H-BYTE ON CARRY
F8A3:	D0 04		RNE	GETFMT	
F8A5:	A0 80	ERP	LDY	#S8C	SUBSTITUTE S8C FOR INVALID OPS
F8A7:	A9 00		LDA	#S0	SET PRINT FORMAT INDEX TO 0
F8A9:	AA	GETFMT	TAX		
F8AA:	BD A6 F9		LDA	FM2,X	INDEX INTO PRINT FORMAT TABLE
F8AD:	85 2E		STA	FORMAT	SAVE FOR ADR FIELD FORMATTING
F8AF:	29 03	*	AND	#S03	MASK FOR 2-BIT LENGTH
					(P=1 BYTE, 1=2 BYTE, 2=3 BYTE)
F8B1:	85 2F		STA	LENGTH	
F8B3:	93		TYA		OPCODE
F8B4:	29 8F		AND	#S8F	MASK FOR 1XXX1010 TEST
F8B6:	AA		TAX		SAVE IT
F8B7:	98		TYA		OPCODE TO A AGAIN
F8B8:	A0 03		LDY	#S03	
F8BA:	E0 8A		CPX	#S8A	
F8BC:	F0 08		BEQ	MNNDX3	
F8BE:	4A	MNNDX1	LSR	A	
F8BF:	90 08		BCC	MNNDX3	FORM INDEX INTO MNEMONIC TABLE
F8C1:	4A		LSR	A	
F8C2:	4A	MNNDX2	LSR	A	1) 1XXX1010=>00101XXX
F8C3:	09 20		ORA	#S20	2) XXXYYY01=>00111XXX
F8C5:	88		DEY		3) XXXYYY10=>00110XXX
F8C6:	D0 FA		BNE	MNNDX2	4) XXXYYY100=>00100XXX
F8C8:	C8		INY		5) XXXXX000=>000XXXXX
F8C9:	88	MNNDX3	DEY		
F8CA:	D0 F2		BNE	MNNDX1	
F8CC:	60		RTS		
F8CD:	FF FF FF		DFB	\$FF,\$FF,\$FF	
F8D0:	20 82 F8	INSTDSP	JSR	INSDS1	GEN FMT, LEN BYTES
F8D3:	48		PHA		SAVE MNEMONIC TABLE INDEX
F8D4:	B1 3A	PRNTOP	LDA	(PCL),Y	
F8D6:	20 DA FD		JSR	PRBYTE	
F8D9:	A2 01		LDX	#S01	PRINT 2 BLANKS
F8DB:	20 4A F9	PRNTBL	JSR	PRBL2	
F8DE:	C4 2F		CPY	LENGTH	PRINT INST (1-3 BYTES)
F8E0:	C8		INY		IN A 12 CHR FIELD
F8E1:	90 F1		BCC	PRNTOP	
F8E3:	A2 03		LDX	#S03	CHAR COUNT FOR MNEMONIC PRINT
F8E5:	C0 04		CPY	#S04	

F8E7: 90 F2		BCC PRNTBL	
F8E9: 68		PLA	RECOVER MNEMONIC INDEX
F8EA: A8		TAY	
F8EB: B9 C0 F9		LDA MNEM1,Y	
F8EE: 85 2C		STA LMNEM	FETCH 3-CHAR MNEMONIC
F8F0: B9 00 FA		LDA MNEMR,Y	(PACKED IN 2-BYTES)
F8F3: 85 2D		STA RMNEM	
F8F5: A9 00	PRMN1	LDA #S00	
F8F7: A0 05		LDY #S05	
F8F9: 06 2D	PRMN2	ASL RMNEM	SHIFT 5 BITS OF
F8FB: 26 2C		ROL LMNEM	CHARACTER INTO A
F8FD: 2A		ROL A	(CLEARS CARRY)
F8FE: 88		DEY	
F8FF: D0 F8		BNE PRMN2	
F901: 69 BF		ADC #SBF	ADD "2" OFFSET
F903: 20 ED FD		JSR COUT	OUTPUT A CHAR OF MNEM
F906: CA		DEX	
F907: D0 EC		BNE PRMN1	
F909: 20 48 F9		JSR PRBLNK	OUTPUT 3 BLANKS
F90C: A4 2F		LDY LENGTH	
F90E: A2 06		LDX #S06	CNT FOR 6 FORMAT BITS
F910: E0 03	PRADR1	CPX #S03	
F912: F0 1C		BEQ PRADR5	IF X=3 THEN ADDR.
F914: 06 2E	PRADR2	ASL FORMAT	
F916: 90 0E		BCC PRADR3	
F918: BD B3 F9		LDA CHAR1-1,X	
F91B: 20 ED FD		JSR COUT	
F91E: BD B9 F9		LDA CHAR2-1,X	
F921: F0 03		BEQ PRADR3	
F923: 20 ED FD		JSR COUT	
F926: CA	PRADR3	DEX	
F927: D0 E7		BNE PRADR1	
F929: 60		RTS	
F92A: 88	PRADR4	DEY	
F92B: 30 E7		BMI PRADR2	
F92D: 20 DA FD		JSR PRRYTE	
F930: A5 2E	PRADR5	LDA FORMAT	
F932: C9 E6		CMP #SE8	HANDLE REL ADR MODE
F934: B1 3A		LDA (PCL),Y	SPECIAL (PRINT TARGET,
F936: 90 F2		BCC PRADR4	NOT OFFSET)
F938: 20 56 F9	RELADR	JSR PCADJ3	
F93B: AA		TAX	PCL,PCH+OFFSET+1 TO A,Y
F93C: E8		INX	
F93D: D0 01		BNE PRNTYX	+1 TO Y,X
F93F: C8		INY	
F940: 98	PRNTYX	TYA	
F941: 20 DA FD	PRNTAX	JSR PRBYTE	OUTPUT TARGET ADR
F944: 8A	PRNTX	TXA	OF BRANCH AND RETURN
F945: 4C DA FD		JMP PRBYTE	
F948: A2 03	PRBLNK	LDX #S03	BLANK COUNT
F94A: A9 A0	PRBL2	LDA #SA0	LOAD A SPACE
F94C: 20 ED FD	PRBL3	JSR COUT	OUTPUT A BLANK
F94F: CA		DEX	
F950: D0 F8		BNE PRBL2	LOOP UNTIL COUNT=0
F952: 60		RTS	
F953: 38	PCADJ	SEC	0=1-BYTE,1=2-BYTE,
F954: A5 2F	PCADJ2	LDA LENGTH	2=3-BYTE
F956: A4 3B	PCADJ3	LDY PCH	
F958: AA		TAX	TEST DISPLACEMENT SIGN
F959: 10 01		BPL PCADJ4	(FOR REL BRANCH)
F95B: 88		DEY	EXTEND NEG BY DECR PCH
F95C: 65 3A	PCADJ4	ADC PCL	
F95E: 90 01		BCC RTS2	PCL+LENGTH(OR DISPL)+1 TO A
F960: C8		INY	CARRY INTO Y (PCH)
F961: 60	RTS2	RTS	
*		FMT1 BYTES:	XXXXXXXXY0 INSTRS
*		IF Y=0	THEN LEFT HALF BYTE
*		IF Y=1	THEN RIGHT HALF BYTE
*			(X=INDEX)
F962: 04 20 54			
F965: 30 0D	FMT1	DFB	\$04,\$20,\$54,\$30,\$0D
F967: 80 04 90			
F96A: 03 22		DFB	\$80,\$04,\$90,\$03,\$22
F96C: 54 33 0D			
F96F: 80 04		DFB	\$54,\$33,\$0D,\$80,\$04
F971: 90 04 20			
F974: 54 33		DFB	\$90,\$04,\$20,\$54,\$33
F976: 0D 80 04			
F979: 90 04		DFB	\$0D,\$80,\$04,\$90,\$04
F97B: 20 54 3B			
F97E: 0D 80		DFB	\$20,\$54,\$3B,\$0D,\$80
F980: 04 90 00			
F983: 22 44		DFB	\$04,\$90,\$00,\$22,\$44
F985: 33 0D C8			
F988: 44 00		DFB	\$33,\$0D,\$C8,\$44,\$00

F98A:	11	22	44		DFB	\$11,\$22,\$44,\$33,\$0D
F98D:	33	0D				
F98F:	C8	44	A9		DFB	SC8,\$44,\$A9,\$01,\$22
F992:	01	22				
F994:	44	33	0D		DFB	\$44,\$33,\$0D,\$80,\$04
F997:	80	04				
F999:	90	01	22		DFB	\$90,\$01,\$22,\$44,\$33
F99C:	44	33				
F99E:	0D	80	04		DFB	\$0D,\$80,\$04,\$90
F9A1:	90					
F9A2:	26	31	87		DFB	\$26,\$31,\$87,\$9A ZZZXXY01 INSTR'S
F9A5:	9A				DFB	\$00 ERR
F9A6:	00			FMT2	DFB	\$21 IMM
F9A7:	21				DFB	\$81 Z-PAGE
F9A8:	81				DFB	\$82 ABS
F9A9:	82				DFB	\$00 IMPLIED
F9AA:	00				DFB	\$00 ACCUMULATOR
F9AB:	00				DFB	\$59 (ZPAG,X)
F9AC:	59				DFB	\$4D (ZPAG),Y
F9AD:	4D				DFB	\$91 ZPAG,X
F9AE:	91				DFB	\$92 ABS,X
F9AF:	92				DFB	\$86 ABS,Y
F9B0:	8E				DFB	\$4A (ABS)
F9B1:	4A				DFB	\$85 ZPAG,Y
F9B2:	85				DFB	\$9D RELATIVE
F9B3:	9D					
F9B4:	AC	A9	AC			
	A3	A8	A4			
				CHAR1	ASC	" ,) , # (\$ "
F9BA:	D9	00	D8			
F9BD:	A4	A4	00		DFB	SD9,\$00,\$D8,\$A4,\$A4,\$00
				CHAR2		"Y",0,"XSS",0
				*CHAR2:		MNEML IS OF FORM:
				*	(A)	XXXXX000
				*	(B)	XXXXY100
				*	(C)	1XXX1010
				*	(D)	XXXXYY10
				*	(E)	XXXXYY01
				*		(X=INDEX)
F9C0:	1C	8A	1C		DFB	\$1C,\$8A,\$1C,\$23,\$5D,\$8B
F9C3:	23	5D	8B			
F9C6:	1B	A1	9D		DFB	\$1B,\$A1,\$9D,\$8A,\$1D,\$23
F9C9:	8A	1D	23			
F9CC:	9D	8B	1D		DFB	\$9D,\$8B,\$1D,\$A1,\$00,\$29
F9CF:	A1	00	29			
F9D2:	19	AE	69		DFB	\$19,\$AE,\$69,\$A8,\$19,\$23
F9D5:	A8	19	23			
F9D8:	24	53	1B		DFB	\$24,\$53,\$1B,\$23,\$24,\$53
F9DB:	23	24	53		DFB	\$19,\$A1 (A) FORMAT ABOVE
F9DE:	19	A1				
F9E0:	00	1A	5B		DFB	\$00,\$1A,\$5B,\$5B,\$A5,\$69
F9E3:	5B	A5	69		DFB	\$24,\$24 (R) FORMAT
F9E6:	24	24				
F9E8:	AE	AE	A8		DFB	\$AE,\$AE,\$A8,\$AD,\$29,\$00
F9EB:	AD	29	00		DFB	\$7C,\$00 (C) FORMAT
F9EE:	7C	00				
F9F0:	15	9C	6D		DFB	\$15,\$9C,\$6D,\$9C,\$A5,\$69
F9F3:	9C	A5	69		DFB	\$29,\$53 (D) FORMAT
F9F6:	29	53				
F9F8:	84	13	34		DFB	\$84,\$13,\$34,\$11,\$A5,\$69
F9FB:	11	A5	69		DFB	\$23,\$A0 (E) FORMAT
F9FE:	23	A0				
FA00:	D8	62	5A		DFB	\$D8,\$62,\$5A,\$48,\$26,\$62
FA03:	48	26	62			
FA06:	94	88	54		DFB	\$94,\$88,\$54,\$44,\$C8,\$54
FA09:	44	C8	54			
FA0C:	68	44	E8		DFB	\$68,\$44,\$E8,\$94,\$00,\$B4
FA0F:	94	00	B4			
FA12:	08	84	74		DFB	\$08,\$84,\$74,\$B4,\$28,\$6E
FA15:	B4	28	6E			
FA18:	74	F4	CC		DFB	\$74,\$F4,\$CC,\$4A,\$72,\$F2
FA1B:	4A	72	F2		DFB	\$A4,\$8A (A) FORMAT
FA1E:	A4	8A				
FA20:	00	AA	A2		DFB	\$00,\$AA,\$A2,\$A2,\$74,\$74
FA23:	A2	74	74		DFB	\$74,\$72 (B) FORMAT
FA26:	74	72				
FA28:	44	68	B2		DFB	\$44,\$68,\$E2,\$32,\$B2,\$00
FA2B:	32	B2	00		DFB	\$22,\$00 (C) FORMAT
FA2E:	22	00				
FA30:	1A	1A	26		DFB	\$1A,\$1A,\$26,\$26,\$72,\$72
FA33:	26	72	72		DFB	\$8E,\$C8 (D) FORMAT
FA36:	88	C8				
FA38:	C4	CA	26		DFB	SC4,\$CA,\$26,\$4E,\$44,\$44
FA3B:	48	44	44			
FA3E:	A2	C8			DFB	SA2,\$C8 (E) FORMAT

FA40: FF FF FF		DFB	\$FF,\$FF,\$FF	
FA43: 20 D0 F8	STEP	JSR	INSTDSP	DISASSEMBLE ONE INST
FA46: 68		PLA		AT (PCL,H)
FA47: 85 2C		STA	RTNL	ADJUST TO USER
FA49: 68		PLA		STACK. SAVE
FA4A: 85 2D		STA	RTNH	RTN ADR.
FA4C: A2 08		LDX	#S08	
FA4E: 8D 10 F8	XQINIT	LDA	INITBL-1,X	INIT XEQ AREA
FA51: 95 3C		STA	XQT,X	
FA53: CA		DEX		
FA54: D0 F8		RNE	XQINIT	
FA56: A1 3A		LDA	(PCL,X)	USER OPCODE BYTE
FA58: F0 42		BEO	XRRK	SPECIAL IF BRRAK
FA5A: A4 2F		LDY	LENGTH	LEN FROM DISASSEMBLY
FA5C: C9 20		CMP	#S20	
FA5E: F0 59		REQ	XJSR	HANDLE JSR, PTS, JMP,
FA60: C9 60		CMP	#S60	JMP (), RTI SPECIAL
FA62: F0 45		BEQ	XRTS	
FA64: C9 4C		CMP	#S4C	
FA66: F0 5C		SEC	XJMP	
FA68: C9 6C		CMP	#S6C	
FA6A: F0 59		BEQ	XJMPAT	
FA6C: C9 40		CMP	#S40	
FA6E: F0 35		BEQ	XRTI	
FA70: 29 1F		AND	#S1F	
FA72: 49 14		EOR	#S14	
FA74: C9 04		CMP	#S04	COPY USER INST TO XEQ AREA
FA76: F0 02		BEQ	XQ2	WITH TRAILING NOPS
FA78: B1 3A	XQ1	LDA	(PCL),Y	CHANGE REL BRANCH
FA7A: 99 3C 00	XQ2	STA	XQTNZ,Y	DISP TO 4 FOR
FA7D: 88		DEY		JMP TO BRANCH OR
FA7E: 10 F8		BPL	XQ1	NBRANCH FROM XEQ.
FA80: 20 3F FF		JSR	RESTORE	RESTORE USER REG CONTENTS.
FA83: 4C 3C 00		JMP	XQTNZ	XEQ USER OP FROM RAM
FA86: 85 45	IRQ	STA	ACC	(RETURN TO NBRANCH)
FA88: 68		PLA		
FA89: 48		PHA		**IRQ HANDLER
FA8A: 0A		ASL	A	
FA8B: 0A		ASL	A	
FA8C: 0A		ASL	A	
FA8D: 30 03		BMI	BREAK	TEST FOR BREAK
FA8F: 6C FE 03		JMP	(IRQLOC)	USER ROUTINE VECTOR IN RAM
FA92: 28	BREAK	PLP		
FA93: 20 4C FF		JSR	SAV1	SAVE REG'S ON BREAK
FA96: 68		PLA		INCLUDING PC
FA97: 85 3A		STA	PCL	
FA99: 68		PLA		
FA9A: 85 3B		STA	PCH	
FA9C: 20 82 F8	XBRK	JSR	INSDS1	PRINT USER PC.
FA9F: 20 DA FA		JSR	RGDSP1	AND REG'S
FAA2: 4C 65 FF		JMP	MON	GO TO MONITOR
FAA5: 18	XRTI	CLC		
FAA6: 68		PLA		SIMULATE RTI BY EXPECTING
FAA7: 85 48		STA	STATUS	STATUS FROM STACK, THEN RTS
FAA9: 68	XRTS	PLA		RTS SIMULATION
FAAA: 85 3A		STA	PCL	EXTRACT PC FROM STACK
FAAC: 68		PLA		AND UPDATE PC BY 1 (LEN=0)
FAAD: 85 3B	PCINC2	STA	PCH	
FAAF: A5 2F	PCINC3	LDA	LENGTH	UPDATE PC BY LEN
FAB1: 20 56 F9		JSR	PCADJ3	
FAB4: 84 3B		STY	PCH	
FAB6: 18		CLC		
FAB7: 90 14		RCC	NEWPCL	
FAB9: 18		CLC		
FABA: 20 54 F9	XJSR	JSR	PCADJ2	UPDATE PC AND PUSH
FABD: AA		TAX		ONTO STACK FOR
FABE: 98		TYA		JSR SIMULATE
FABF: 48		PHA		
FAC0: 8A		TXA		
FAC1: 48		PHA		
FAC2: A0 02		LDY	#S02	
FAC4: 18	XJMP	CLC		
FAC5: B1 3A	XJMPAT	LDA	(PCL),Y	
FAC7: AA		TAX		LOAD PC FOR JMP,
FAC8: 88		DEY		(JMP) SIMULATE.
FAC9: B1 3A		LDA	(PCL),Y	
FACB: 86 3B		STX	PCH	
FACD: 85 3A	NEWPCL	STA	PCL	
FACF: B0 F3		BCS	XJMP	
FAD1: A5 2D	RTNJ4P	LDA	RTNH	
FAD3: 48		PHA		
FAD4: A5 2C		LDA	RTNL	
FAD6: 48		PHA		
FAD7: 20 8E FD	REGDSP	JSR	CROUT	DISPLAY USER REG
FADA: A9 45	RGDSP1	LDA	#ACC	CONTENTS WITH
FADC: 85 40		STA	A3L	LABELS

FADE: A9 00		LDA #ACC/256	
FAE0: 85 41		STA A3H	
FAE2: A2 FB		LDX #SFB	
FAE4: A9 A0	RDSP1	LDA #SA0	
FAE6: 20 ED FD		JSR COUT	
FAE9: BD 1E FA		LDA RTBL-SFB,X	
FAEC: 20 ED FD		JSR COUT	
FAEF: A9 BD		LDA #SBD	
FAF1: 20 ED FD		JSR COUT	
FAF4: B5 4A		LDA ACC+5,X	
FAF6: 20 DA FD		JSR PFBYTE	
FAF9: E8		INX	
FAFA: 30 E8		BMI RDSP1	
FAFC: 60		RTS	
FAFD: 18	BRANCH	CLC	BRANCH TAKEN,
FAFE: A0 01		LDY #S01	ADD LEN+2 TO PC
FB00: B1 3A		LDA (PCL),Y	
FB02: 20 56 F9		JSR PCADJ3	
FB05: 85 3A		STA PCL	
FB07: 98		TYA	
FB08: 38		SEC	
FB09: B0 A2		BCC PCINC2	
FB0B: 20 4A FF	NRPNCB	JSR SAVE	NORMAL RETURN AFTER
FB0E: 38		SEC	XEQ USER OF
FB0F: B0 9E		BCC PCINC3	GO UPDATE PC
FB11: EA	INITBL	NOP	
FB12: EA		NOP	DUMMY FILL FOR
FB13: 4C 0B FB		JMP NRRNCB	XEQ AREA
FB16: 4C FD FA		JMP BRANCH	
FB19: C1	RTBL	DFB SC1	
FB1A: D8		DFB SD8	
FB1B: D9		DFB SD9	
FB1C: D0		DFB SD0	
FB1D: D3		DFB SD3	
FB1E: AD 70 C0	PREAD	LDA PTPIG	TRIGGER PADDLES
FB21: A0 00		LDY #S00	INIT COUNT
FB23: EA		NOP	COMPENSATE FOR 1ST COUNT
FB24: EA		NOP	
FB25: BD 64 C0	PREAD2	LDA PADDL0,X	COUNT Y-REG EVERY
FB28: 10 04		BPL RTS2D	12 USEC
FB2A: C8		INY	
FB2B: D0 F8		BNE PREAD2	EXIT AT 255 MAX
FB2D: 88		DEY	
FB2E: 60	RTS2D	RTS	
FB2F: A9 00	INIT	LDA #S00	CLR STATUS FOR DERUG
FB31: 85 48		STA STATUS	SOFTWARE
FB33: AD 56 C0		LDA LORES	
FB36: AD 54 C0		LDA LOWSCR	INIT VIDEO MODE
FB39: AD 51 C0	SETTXT	LDA TXTSET	SET FOR TEXT MODE
FB3C: A9 00		LDA #S00	FULL SCREEN WINDOW
FB3E: F0 08		REQ SETWND	
FB40: AD 50 C0	SETGR	LDA TXTCLR	SET FOR GRAPHICS MODE
FB43: AD 53 C0		LDA MIXSET	LOWER 4 LINES AS
FB46: 20 36 F8		JSR CLPTOP	TEXT WINDOW
FB49: A9 14		LDA #S14	
FB4B: 85 22	SETWND	STA WNDTOP	SET FOR 40 COL WINDOW
FB4D: A9 00		LDA #S00	TOP IN A-RFC,
FB4F: 85 20		STA WNDLFT	BTM AT LINE 24
FB51: A9 28		LDA #S28	
FB53: 85 21		STA WNDWOTH	
FB55: A9 18		LDA #S16	
FB57: 85 23		STA WNDRTM	VTAB TO ROW 23
FB59: A9 17		LDA #S17	
FB5B: 85 25	TABV	STA CV	VTABS TO ROW IN A-REG
FB5D: 4C 22 FC		JMP VTAB	
FB60: 20 A4 FE	MULPM	JSR MD1	ABS VAL OF AC AUX
FB63: A0 10	MUL	LDY #S10	INDEX FOR 16 BITS
FB65: A5 50	MUL2	LDA AC1	ACK = AUX + XTND
FB67: 4A		LSR A	TO AC, XTND
FB68: 90 0C		BCC MUL4	IF NO CAPRY,
FB6A: 18		CLC	NO PARTIAL PP0D.
FB6B: A2 FE		LDX #SFE	
FB6D: B5 54	MUL3	LDA XTNDL+2,X	ADD MPLCND (AUX)
FB6F: 75 56		ADC AUXL+2,X	TO PARTIAL PROD
FB71: 95 54		STA XTNDL+2,X	(XTND).
FB73: E8		INX	
FB74: D0 F7		BNE MUL3	
FB76: A2 03	MUL4	LDX #S03	
FB78: 76	MUL5	DFB #S76	
FB79: 50		DFB #S50	
FB7A: CA		DEX	
FB7B: 10 FE		BPL MUL5	
FB7D: 88		DEY	
FB7E: D0 E5		BNE MUL2	
FB80: 60		RTS	

FB81:	20 A4	FB	DIVPM	JSR MD1	AWS VAL OF AC, AUX.
FB84:	A0 10		DIV	LDY #S10	INDEX FOR 16 BITS
FB86:	06 50		DIV2	ASL ACL	
FB88:	26 51			ROL ACH	
FB8A:	26 52			ROL XTNDL	YTMD/AUX
FB8C:	26 53			ROL XTNDH	TO AC.
FB8E:	38			SEC	
FB8F:	A5 52			LDA XTNDL	
FB91:	F5 54			SEC AUXL	MOD TO XTND.
FB93:	AA			TAX	
FB94:	A5 53			LDA XTNDH	
FB96:	E5 55			SEC AUXL	
FB98:	90 06			BCC DIV3	
FB9A:	86 52			STX XTNDL	
FB9C:	85 53			STA XTNDL	
FB9E:	E6 50			INC ACL	
FBA0:	88		DIV3	DEY	
FBA1:	D0 E3			BNE DIV2	
FBA3:	60			RTS	
FBA4:	A0 00	MD1		LDY #S00	PRS VAL OF AC, AUX
FBA6:	84 2F			STY SIGN	WITH RESULT SIGN
FBA8:	A2 54			LDX #AUXL	IN LSB OF SIGN.
FBA A:	20 AF	FE		JSR MD2	
FBAD:	A2 50			LDX #ACL	
FBAF:	B5 01	MD2		LDA LOC1,X	X SPECIFIES AC OR AUX
FBB1:	10 0D			RPL MDRTS	
FBB3:	38			SEC	
FBB4:	98	MD3		TYA	
FBB5:	F5 00			SBC LOC0,X	COMPL SPECIFIED REG
FBB7:	95 00			STA LOC0,X	IF NEG.
FBB9:	98			TYA	
FBB A:	F5 01			SBC LOC1,X	
FBBC:	95 01			STA LOC1,X	
FBBE:	E6 2F			INC SIGN	
FBC0:	60	MDRTS		RTS	
FBC1:	48	BASCALC		PHA	CALC BASE ADR IN BASL,H
FBC2:	4A			LSR A	FOR GIVEN LINE NO.
FBC3:	29 03			AND #S03	0<=LINE NO.<=S17
FBC5:	09 04			ORA #S04	APG=000ABCDE, GENERATE
FBC7:	85 29			STA BASH	BASH=000001CD
FBC9:	68			PLA	AND
FBC A:	29 18			AND #S18	BASL=EABAB000
FBC C:	90 02			BCC BSCLC2	
FBC E:	69 7F			ADC #S7F	
FBD0:	85 28	BSCLC2		STA BASL	
FBD2:	0A			ASL A	
FBD3:	0A			ASL A	
FBD4:	05 28			ORA BASL	
FBD6:	85 28			STA BASL	
FBD8:	60			RTS	
FBD9:	C9 87	BELL1		CMP #S87	BELL CHAR? (CNTRL-G)
FBD B:	D0 12			BNE RTS2E	NO, RETURN
FBD D:	A9 40			LDA #S40	DELAY .01 SECONDS
FBD F:	20 A8	FC		JSR WAIT	
FBE2:	A0 C0			LDY #SC0	
FBE4:	A9 0C	BELL2		LDA #S0C	TOGGLE SPEAKER AT
FBE6:	20 A8	FC		JSR WAIT	1 KHZ FOR .1 SEC.
FBE9:	AD 30	C0		LDA SPKR	
FBE C:	88			DEY	
FBE D:	D0 F5			BNE BELL2	
FBE F:	60	RTS2P		RTS	
FBF0:	A4 24	STOADV		LDY CH	CURSER H INDEX TO Y-REG
FBF2:	91 28			STA (BASL),Y	STOR CHAR IN LINE
FBF4:	E6 24	ADVANCE		INC CH	INCREMENT CURSER H INDEX
FBF6:	A5 24			LDA CH	(MOVE RIGHT)
FBF8:	C5 21			CMP WNDWDTH	BEYOND WINDOW WIDTH?
FBF A:	B0 66			BCC CR	YES 'P TO NEXT LINE
FBF C:	60	PTS3		RTS	NO, RETURN
FBF D:	C9 A0	VIDOUT		CMP #SA0	CONTROL CHAR?
FBF F:	B0 EF			BCC STOADV	NO, OUTPUT IT.
FC01:	A8			TAY	INVERSE VIDEO?
FC02:	10 EC			RPL STOADV	YES, OUTPUT IT.
FC04:	C9 8D			CMP #S8D	CR?
FC06:	F0 5A			BEQ CR	YES.
FC08:	C9 8A			CMP #S8A	LINE FEED?
FC0 A:	F0 5A			BEQ LF	IF SO, DO IT.
FC0 C:	C9 88			CMP #S88	BACK SPACE? (CNTRL-H)
FC0 E:	D0 C9			BNE BELL1	NO, CHECK FOR BELL.
FC10:	C6 24	PS		DEC CH	DECREMENT CURSER H INDEX
FC12:	10 E8			BPL RTS3	IF POS, OK. ELSE MOVE UP
FC14:	A5 21			LDA WNDWDTH	SET CH TO WNDWDTH-1
FC16:	85 24			STA CH	
FC18:	C6 24			DEC CH	(RIGHTMOST SCREEN POS)
FC1 A:	A5 22	UP		LDA WNDTOP	CURSER V INDEX
FC1 C:	C5 25			CMP CV	

FC1E: B0 0B		BCS RTS4	IF TOP LINE THEN RETURN
FC20: C6 25		DEC CV	DPCR CURSER V-INDEX
FC22: A5 25	VTAB	LDA CV	GET CURSER V-INDEX
FC24: 20 C1 FB	VTABZ	JSR PASCALC	GENERATE BASE ADDR
FC27: 65 20		ADC WNDLFT	ADD WINDOW LEFT INDEX
FC29: 85 28		STA BASL	TO BASL
FC2B: 60	RTS4	PTS	
FC2C: 49 C0	ESC1	EOR #S00	ESC?
FC2E: F0 28		RFO HOME	IF SO, DO HOME AND CLEAP
FC30: 69 FD		ADC #SFD	ESC-A OR B CHECK
FC32: 90 C0		BCC ADVANCE	A, ADVANCE
FC34: F0 DA		BEO BS	B, BACKSPACE
FC36: 69 FD		ADC #SFD	ESC-C OR D CHECK
FC38: 90 2C		BCC LF	C, DOWN
FC3A: F0 DE		BEQ UP	D, GO UP
FC3C: 69 FD		ADC #SFD	ESC-E OF F CHECK
FC3E: 90 5C		BCC CLREOL	F, CLEAR TO END OF LINE
FC40: D0 E9		BNE RTS4	NOT F, RETURN
FC42: A4 24	CLREOP	LDY CH	CURSOR H TO Y INDEX
FC44: A5 25		LDA CV	CURSOR V TO A-REGISTER
FC46: 48	CLEOP1	PHA	SAVE CURRENT LINE ON STK
FC47: 20 24 FC		JSR VTABZ	CALC BASE ADDRESS
FC4A: 20 9E FC		JSR CLEOLZ	CLEAR TO EOL, SET CARRY
FC4D: A0 00		LDY #S00	CLEAR FROM H INDEX=0 FOR REST
FC4F: 68		PLA	INCREMENT CURRENT LINE
FC50: 69 00		ADC #S00	(CARRY IS SET)
FC52: C5 23		CMP WNDRTM	DONE TO BOTTOM OF WINDOW?
FC54: 90 F0		BCC CLEOP1	NO, KEEP CLEAPING LINES
FC56: B0 CA		BCC VTAB	YES, TAB TO CURRENT LINE
FC58: A5 22	HOME	LDA WNDTOP	INIT CURSOR V
FC5A: 85 25		STA CV	AND H-INDICES
FC5C: A0 00		LDY #S00	
FC5E: 84 24		STY CH	THEN CLEAR TO END OF PAGE
FC60: F0 E4		BEO CLEOP1	
FC62: A9 00	CR	LDA #S00	CURSOR TO LEFT OF INDEX
FC64: 85 24		STA CH	(PET CURSOR H=0)
FC66: E6 25	LF	INC CV	INCR CURSOR V(DOWN 1 LINE)
FC68: A5 25		LDA CV	
FC6A: C5 23		CMP WNDPTM	OFF SCREFN?
FC6C: 90 B6		BCC VTABZ	NO, SET BASE ADDR
FC6E: C6 25		DEC CV	DECP CURSOR V(BACK TO BOTTOM LINE)
FC70: A5 22	SCROLL	LDA WNDTOP	START AT TOP OF SCRL WNDW
FC72: 48		PHA	
FC73: 20 24 FC		JSR VTABZ	GENERATE BASE ADDRESS
FC76: A5 28	SCRL1	LDA BASL	COPY BASL, H
FC78: 85 2A		STA BAS2L	TO BAS2L, H
FC7A: A5 29		LDA BAS4	
FC7C: 85 2B		STA BAS2H	
FC7E: A4 21		LDY WNDWOTH	INIT Y TO RIGHTMOST INDEX
FC80: 88		DEFY	OF SCROLLING WINDOW
FC81: 68		PLA	
FC82: 69 01		ADC #S01	INCR LINE NUMBER
FC84: C5 23		CMP WNDRTM	DONE?
FC86: B0 0D		BCS SCRL3	YES, FINISH
FC88: 48		PHA	
FC89: 20 24 FC		JSR VTABZ	FORM BASL, H (BASE ADDR)
FC8C: B1 28	SCRL2	LDA (BASL), Y	MOVE A CHR UP ON LINE
FC8E: 91 2A		STA (BAS2L), Y	
FC90: 88		DEY	NEXT CHAR OF LINE
FC91: 10 F9		BPL SCRL2	
FC93: 30 E1		BMI SCRL1	NEXT LINE
FC95: A0 00	SCRL3	LDY #S00	CLEAP BOTTOM LINE
FC97: 20 9E FC		JSR CLEOLZ	SET BASE ADDR FOR BOTTOM LINE
FC9A: B0 86		BCC VTAB	CARRY IS SET
FC9C: A4 24	CLREOL	LDY CH	CURSOR H INDEX
FC9E: A9 A0	CLEOLZ	LDA #SA0	
FCA0: 91 28	CLFOL2	STA (BASL), Y	STORE BLANKS FROM 'HERE'
FCA2: C8		INY	TO END OF LINES (WNDWOTH)
FCA3: C4 21		CPY WNDWOTH	
FCA5: 90 F9		BCC CLEOLZ	
FCA7: 60		PTS	
FCA8: 38	WAIT	SEC	
FCA9: 48	WAIT2	PHA	
FCAA: E9 01	WAIT3	SBC #S01	
FCAC: D0 FC		PNE WAIT3	1.0204 USEC
FCAE: 68		PLA	(13+2712*A+512*A*A)
FCAF: E9 01		SBC #S01	
FCB1: D0 F6		BNE WAIT2	
FCB3: 60		PTS	
FCB4: E6 42	NXTA4	INC A4L	INCR 2-BYTE A4
FCB6: D0 02		PNE NXTA1	AND A1
FCB8: E6 43		INC A4H	
FCBA: A5 3C	NXTA1	LDA A1L	INCP 2-BYTE A1.
FCBC: C5 3E		CMP A2L	
FCBE: A5 3D		LDA A1H	AND COMPARE TO A2

FCC0: E5 3F		SBC A2H	
FCC2: E6 3C		INC AL	(CARRY SET IF >=)
FCC4: D0 02		BNE RTS4R	
FCC6: E6 3D		INC ALH	
FCC8: 60	RTS4R	RTS	
FCC9: A0 43	HEADR	LDY #S4R	WRITE A*256 'LONG 1'
FCCR: 20 DB FC		JSR ZERDLY	HALF CYCLES
FCE2: D0 F9		BNE HEADR	(650 USEC EACH)
FCD0: 69 FE		ADC #SFE	
FCD2: F0 F5		RCS HEADR	THEN A 'SHORT 0'
FCD4: A0 21		LDY #S21	(400 USEC)
FCD6: 20 DB FC	WRBIT	JSR ZERDLY	WRITE TWO HALF CYCLES
FCD9: C8		INY	OF 250 USEC ('0')
FCD A: C8		INY	OR 500 USEC ('0')
FCD B: 88	ZERDLY	DEY	
FCD C: D0 FD		BNE ZERDLY	
FCD E: 90 05		RCC WRTAPE	Y IS COUNT FOR
FCE0: A0 32		LDY #S32	TIMING LOOP
FCE2: 88	ONEDLY	DEY	
FCE3: D0 FD		BNE ONEDLY	
FCE5: AC 20	C0 . WRTAPE	LDY TAPEOUT	
FCE8: A0 2C		LDY #S2C	
FCE A: CA		DEX	
FCE B: 60		RTS	
FCE C: A2 08	RDRYTE	LDX #S08	8 BITS TO READ
FCE E: 48	RDRYT2	PHA	READ TWO TRANSITIONS
FCE F: 20 FA FC		JSP RD2BIT	(FIND EDGE)
FCF2: 68		PLA	
FCF3: 2A		ROL A	NEXT BIT
FCF4: A0 3A		LDY #S3A	COUNT FOR SAMPLES
FCF6: CA		DEX	
FCF7: D0 F5		BNE RDRYT2	
FCF9: 60		RTS	
FCFA: 20 FD FC	RD2BIT	JSR RD3IT	
FCFD: 88	RDPIT	DEY	DECR Y UNTIL
FCFE: AD 60 C0		LDA TAPEIN	TAPE TRANSITION
FD01: 45 2F		EOR LASTIN	
FD03: 10 F8		RPL RDBIT	
FD05: 45 2F		EOR LASTIN	
FD07: 85 2F		STA LASTIN	
FD09: C0 80		CPY #S80	SET CARRY ON Y-REG.
FD0B: 60		RTS	
FD0C: A4 24	RDKEY	LDY CH	
FD0E: B1 28		LDA (FASL),Y	SET SCREEN TO FLASH
FD10: 48		PHA	
FD11: 29 3F		AND #S3F	
FD13: 09 40		ORA #S40	
FD15: 91 28		STA (FASL),Y	
FD17: 68		PLA	
FD18: 6C 38 00		JSP (KSWL)	GO TO USER KEY-IN
FD1E: E6 4E	KEYIN	INC ENDL	
FD1D: D0 02		BNE KEYIN2	INCR RND NUMBER
FD1F: E6 4F		INC RNDH	
FD21: 2C 00 C0	KEYIN2	BIT RND	KEY DOWN?
FD24: 10 F5		BPL KEYIN	LOOP
FD26: 91 28		STA (BASL),Y	REPLACE FLASHING SCREEN
FD28: AD 00 C0		LDA RSD	GET KEYCODE
FD2B: 2C 10 C0		RIT KPDSTPP	CLR KEY STORE
FD2E: 60		RTS	
FD2F: 20 0C FD	ESC	JSR PDKEY	GET KEYCODE
FD32: 20 2C FC		JSR ESC1	HANDLE ESC FUNC.
FD35: 20 0C FD	PDCHAR	JSP PDKEY	READ KEY
FD38: C9 93		CMP #S93	ESC?
FD3A: F0 F3		BEQ ESC	YES, DON'T RETURN
FD3C: 60		RTS	
FD3D: A5 32	NOTCR	LDA INVFLG	
FD3F: 48		PHA	
FD40: A9 FF		LDA #SFF	
FD42: 85 32		STA INVFLG	ECHO USER LINE
FD44: BD 00 02		LDA IN,X	NON INVERSE
FD47: 20 ED FD		JSR COUT	
FD4A: 68		PLA	
FD4B: 85 32		STA INVFLG	
FD4D: BD 00 02		LDA IN,X	
FD50: C9 88		CMP #S88	CHECK FOR EDIT KEYS
FD52: F0 1D		BEQ BCKSPC	BS, CTRL-X.
FD54: C9 98		CMP #S98	
FD56: F0 0A		BEQ CANCEL	
FD58: E0 F8		CPX #SF8	MARGIN?
FD5A: 90 03		RCC NOTCR1	
FD5C: 20 3A FF		JSR BELL	YES, SOUND BELL
FD5F: E8	NOTCR1	INX	ADVANCE INPUT INDEX
FD60: D0 13		BNE NXTCHAR	
FD62: A9 DC	CANCEL	LDA #SDC	BACKSLASH AFTER CANCELLED LINE
FD64: 20 ED FD		JSR COUT	

FD67:	20 8E	FD	GETLNZ	JSR	CROUT	OUTPUT CR
FD6A:	A5 33		GETLN	LDA	PROMPT	
FD6C:	20 ED	FD		JSR	COUT	OUTPUT PROMPT CHAP
FD6F:	A2 01			LDX	#S01	INIT INPUT INDEX
FD71:	8A		RCKSPC	TXA		WILL BACKSPACE TO 0
FD72:	F0 F3			BEQ	GETLNZ	
FD74:	CA			DEX		
FD75:	20 35	FD	NXTCHAR	JSR	PDCHAR	
FD78:	C9 95			CMP	#PICK	USE SCREEN CHAR
FD7A:	D0 02			BNE	CAPTST	FOR CTRL-U
FD7C:	B1 28			LDA	(BASL),Y	
FD7E:	C9 E0		CAPTST	CMP	#SF0	
FD80:	90 02			BCC	ADDINP	CONVERT TO CAPS
FD82:	29 DF			AND	#SDF	
FD84:	9D 00	02	ADDINP	STA	IN,X	ADD TO INPUT BUF
FD87:	C9 8D			CMP	#S8D	
FD89:	D0 B2			BNE	NOTCR	
FD8B:	20 9C	FC		JSR	CLPEOL	CLR TO EOL IF CR
FD8E:	A9 8D		CROUT	LDA	#S8D	
FD90:	D0 5B			BNE	COUT	
FD92:	A4 3D		PFA1	LDY	A1H	PRINT CR,A1 IN HEX
FD94:	A6 3C			LDX	A1L	
FD96:	20 8E	FD	PRYX2	JSR	CROUT	
FD99:	20 40	F9		JSP	PRJTYX	
FD9C:	A0 00			LDY	#S00	
FD9E:	A9 AD			LDA	#SAD	PRINT '-'
FDA0:	4C ED	FD		JMP	COUT	
FDA3:	A5 3C		XAM8	LDA	A1L	
FDA5:	09 07			CRA	#S07	SET TO FINISH AT
FDA7:	85 3E			STA	A2L	MOD 8=7
FDA9:	A5 3D			LDA	A1H	
FDAB:	85 3F			STA	A2H	
FDAD:	A5 3C		MODRCHK	LDA	A1L	
FDAF:	29 07			AND	#S07	
FDB1:	D0 03			BNE	DATAOUT	
FDB3:	20 92	FD	XAM	JSR	PRA1	
FDB6:	A9 A0		DATAOUT	LDA	#SA0	
FDB8:	20 FD	FD		JSR	COUT	OUTPUT BLANK
FDBB:	B1 3C			LDA	(A1L),Y	
FDBD:	20 DA	FD		JSR	PRBYTE	OUTPUT BYTE IN HEX
FDC0:	20 BA	FC		JSR	NXTA1	
FDC3:	90 E8			BCC	MODRCHK	CHECK IF TIME TO,
FDC5:	60		RTS4C	RTS		PRINT ADDR
FDC6:	4A		XAMPM	LSR	A	DETERMINE IF MON
FDC7:	90 EA			BCC	XAM	MODE IS XAM
FDC9:	4A			LSR	A	ADD, OR SUB
FDCA:	4A			LSR	A	
FDCB:	A5 3E			LDA	A2L	
FDCD:	90 02			BCC	ADD	
FDCF:	49 FF			EOR	#SFF	SUB: FORM 2'S COMPLEMENT
FDD1:	65 3C		ADD	ADC	A1L	
FDD3:	48			PHA		
FDD4:	A9 BD			LDA	#S8D	
FDD6:	20 ED	FD		JSR	COUT	PRINT '=', THEN RESULT
FDD9:	68			PLA		
FDDA:	48		PRBYTE	PHA		PRINT BYTE AS 2 HEX
Fddb:	4A			LSR	A	DIGITS, DESTROYS A-REG
FDDC:	4A			LSR	A	
FDDD:	4A			LSR	A	
FDDE:	4A			LSR	A	
FDDF:	20 E5	FD		JSR	PRHEXZ	
FDE2:	68			PLA		
FDE3:	29 0F		PRHEX	AND	#S0F	PRINT HEX DIG IN A-REG
FDE5:	09 B0		PRHEXZ	ORA	#S20	LSB'S
FDE7:	C9 BA			CMP	#SBA	
FDE9:	90 02			BCC	COUT	
FDEB:	69 06			ADC	#S06	
FDED:	6C 36	00	COUT	JMP	(CSFL)	VECTOR TO USER OUTPUT ROUTINE
FDF0:	C9 A0		COUT1	CMP	#SA0	
FDF2:	90 02			BCC	COUTZ	DON'T OUTPUT CTRL'S INVERSE
FDF4:	25 32			AND	INVFLG	MASK WITH INVERSE FLAG
FDF6:	84 35		COUTZ	STY	YSAV1	SAV Y-REG
FDF8:	48			PHA		SAV A-REG
FDF9:	20 FD	FB		JSR	VIDOUT	OUTPUT A-REG AS ASCII
FDFC:	68			PLA		RESTORE A-REG
FDFD:	A4 35			LDY	YSAV1	AND Y-RFG
FDFE:	60			RIS		THEN RETURN
FE00:	C6 34		BL1	DEC	YSAV	
FE02:	F0 9F			BEQ	XAM8	
FE04:	CA		BLANK	DEX		BLANK TO MON
FE05:	D0 16			BNE	SETMDZ	AFTER BLANK
FE07:	C9 BA			CMP	#SBA	DATA STORE MODE?
FE09:	D0 BB			BNE	XAMPM	NO, XAM, ADD OR SUB
FE0B:	85 31		STOR	STA	MODE	KEEP IN STORE MODE
FE0D:	A5 3E			LDA	A2L	

FE0F: 91 40		STA (A3L),Y	STORE AS LOW BYTE AS (A3)
FE11: E6 40		INC A3L	
FE13: D0 02		RNE RTS5	INCR A3, RETURN
FE15: E6 41		INC A3H	
FE17: 60	RTS5	RTS	
FE18: A4 34	SETMODE	LDY YSAV	SAVE CONVERTED ':', '+',
FE1A: B9 FF 01		LDA IN-1,Y	'-', '.', AS MODE.
FE1D: 85 31	SETMDZ	STA MODF	
FE1F: 60		RTS	
FE20: A2 01	LI	LDX #S01	
FE22: B5 3E	LT2	LDA A2L,X	COPY A2 (2 BYTES) TO
FE24: 95 42		STA A4L,X	A4 AND A5
FE26: 95 44		STA A5L,X	
FE28: CA		DEX	
FE29: 10 F7		RPL LT2	
FE2B: 60		RTS	
FE2C: B1 3C	MOVE	LDA (A1L),Y	MOVE (A1 TO A2) TO
FE2E: 91 42		STA (A4L),Y	(A4)
FE30: 20 B4 FC		JSR NXTA4	
FE33: 90 F7		RCC MOVE	
FE35: 60		RTS	
FE36: B1 3C	VFY	LDA (A1L),Y	VERIFY (A1 TO A2) WITH
FE38: D1 42		CMP (A4L),Y	(A4)
FE3A: F0 1C		BEO VFYOK	
FE3C: 20 92 FD		JSR PR11	
FE3F: B1 3C		LDA (A1L),Y	
FE41: 20 DA FD		JSR PRYTE	
FE44: A9 A0		LDA #SA0	
FE46: 20 ED FD		JSR COUT	
FE49: A9 A8		LDA #SA8	
FE4B: 20 ED FD		JSR COUT	
FE4E: B1 42		LDA (A4L),Y	
FE50: 20 DA FD		JSP PRYTE	
FE53: A9 A9		LDA #SA9	
FE55: 20 ED FD		JSR COUT	
FE58: 20 B4 FC	VFYOK	JSR NXTA4	
FE5B: 90 D9		RCC VFY	
FE5D: 60		RTS	
FE5E: 20 75 FE	LIST	JSP A1PC	MOVE A1 (2 BYTES) TO
FE61: A9 14		LDA #S14	PC IF SPEC'D AND
FE63: 48	LIST2	PIA	DISSEMBLE 20 INSTRS
FE64: 20 D0 F8		JSR INSTDSP	
FE67: 20 53 F9		JSP PCADJ	ADJUST PC EACH INSTR
FE6A: 85 3A		STA PCL	
FE6C: 84 3B		STY PCH	
FE6E: 68		PIA	
FE6F: 38		SEC	
FE70: E9 01		SBC #S01	NEXT OF 20 INSTRS
FE72: D0 EF		RNE LIST2	
FE74: 60		RTS	
FE75: 8A	A1PC	TXA	IF USFR SPEC'D ADR
FE76: F0 07		REQ A1PCRTS	COPY FROM A1 TO PC
FE78: B5 3C	A1PCLP	LDA A1L,X	
FE7A: 95 3A		STA PCL,X	
FE7C: CA		DEX	
FE7D: 10 F9		FPL A1PCIP	
FE7F: 60	A1PCRTS	RTS	
FE80: A0 3F	SETINV	LDY #S3F	SET FOR INVERSE VID
FE82: D0 02		BNE SETIFLG	VIA COUT1
FE84: A0 FF	SETNORM	LDY #SFF	SET FOR NORMAL VID
FE86: 84 32	SETIFLG	STY INVFLG	
FE88: 60		RTS	
FE89: A9 00	SETK3D	LDA #S00	SIMULATE PORT #0 INPUT
FE8B: 85 3E	INPORT	STA A2L	SPECIFIED (KEYIN ROUTINE)
FE8D: A2 36	INPRT	LDX #KSWL	
FE8F: A0 1B		LDY #KEYIN	
FE91: D0 08		RNE IOPRT	
FE93: A9 00	SETVID	LDA #S00	SIMULATE PORT #0 OUTPUT
FE95: 85 3E	OUTPORT	STA A2L	SPECIFIED (COUT1 ROUTINE)
FE97: A2 36	OUTPRT	LDX #CSWL	
FE99: A0 F0		LDY #COUT1	
FE9B: A5 3E	IOPRT	LDA A2L	SET PAM IN/OUT VECTORS
FE9D: 29 0F		AND #S0F	
FE9F: F0 06		BEQ IOPRT1	
FEA1: 09 C0		OPA #IGADR/256	
FEA3: A0 00		LDY #S00	
FEA5: F0 02		REQ IOPRT2	
FEA7: A9 FD	IOPRT1	LDA #COUT1/256	
FEA9: 94 00	IOPRT2	STY LOC0,X	
FEAB: 95 01		STA LOC1,X	
FEAD: 60		PTS	
FEAE: EA		NOP	
FEAF: EA		NOP	
FEB0: 4C 00 E0	XBASIC	JMP BASIC	TO BASIC WITH SCRATCH
FEB3: 4C 03 E0	BASCONT	JMP BASIC2	CONTINUE BASIC

FEB6: 20 75 FE	GO	JSR	ALPC	ADR TO PC IF SPEC'D
FEB9: 20 3F FF		JSR	RESTORE	RESTORE META REGS
FEBC: 6C 3A 00		JMP	(PCL)	GO TO USER SUBR
FEBF: 4C D7 FA	REGZ	JMP	REGDSP	TO REG DISPLAY
FEC2: C6 34	TPACF	DEC	YSAV	
FEC4: 20 75 FE	STEPZ	JSR	ALPC	ADR TO PC IF SPEC'D
FEC7: 4C 43 FA		JMP	STFP	TAKE ONE STEP
FECA: 4C F8 03	USP	JMP	USRADR	TO USP SUBR AT USRADR
FECD: A9 40	WRITE	LDA	#S40	
FECF: 20 C9 FC		JSR	HEADR	WRITE 10-SEC HEADER
FED2: A0 27		LDY	#S27	
FED4: A2 00	WR1	LDX	#S00	
FED6: 41 3C		EOP	(ALL,X)	
FED8: 48		PHA		
FED9: A1 3C		LDA	(ALL,X)	
FEDB: 20 ED FE		JSR	WRPYTE	
FEDE: 20 BA FC		JSR	NXTA1	
FEEL: A0 1D		LDY	#S10	
FEE3: 68		FLA		
FEE4: 90 EE		BCC	WR1	
FEE6: A0 2E		LDY	#S22	
FEE8: 20 ED FE		JSR	WRPYTE	
FEEB: F0 40		BEC	BELL	
FEED: A2 10	WRBYTE	LDX	#S10	
FEED: 0A	WRBYT2	ASL	A	
FEF0: 20 D6 FC		JSR	WRBIT	
FEF3: D0 FA		BNE	WRBYT2	
FEF5: 60		RTS		
FEF6: 20 00 FE	CRMON	JSR	BL1	HANDLE CR AS PLANK
FEF9: 68		PLA		THEN POP STACK
FEFA: 68		PLA		AND RTN TO MON
FEFB: D0 6C		BNE	MONZ	
FEFD: 20 FA FC	READ	JSR	RD2BIT	FIND TAPEIN EDGE
FF00: A9 16		LDA	#S16	
FF02: 20 C9 FC		JSR	HEADR	DELAY 3.5 SECONDS
FF05: 85 2E		STA	CHKSUM	INIT CHKSUM=SFF
FF07: 20 FA FC		JSR	RD2BIT	FIND TAPEIN EDGE
FF0A: A0 24	RD2	LDY	#S24	LOOK FOR SYNC BIT
FF0C: 20 FD FC		JSR	RD2BIT	(SHORT 0)
FF0F: B0 F9		BCS	RD2	LOOP UNTIL FOUND
FF11: 20 FD FC		JSR	RD2BIT	SKIP SECOND SYNC H-CYCLE
FF14: A0 3B		LDY	#S35	INDEX FOR 0/1 TEST
FF16: 20 EC FC	RD3	JSR	RDBYTE	READ A BYTE
FF19: 81 3C		STA	(ALL,X)	STORE AT (A1)
FF1B: 45 2E		EOR	CHKSUM	
FF1D: 85 2E		STA	CHKSUM	UPDATE RUNNING CHKSUM
FF1F: 20 3A FC		JSR	NXTA1	INCR A1, COMPARE TO A2
FF22: A0 35		LDY	#S35	COMPENSATE 0/1 INDEX
FF24: 90 F0		BCC	RD3	LOOP UNTIL DONE
FF26: 20 EC FC		JSR	RDBYTE	READ CHKSUM BYTE
FF29: C5 2E		CMP	CHKSUM	
FF2B: F0 0D		BEC	BELL	GOOD, SOUND BELL AND RETURN
FF2D: A9 C5	PPERR	LDA	#SC5	
FF2F: 20 ED FD		JSR	COUT	PRINT "ERR", THEN BELL
FF32: A9 D2		LDA	#SD2	
FF34: 20 ED FD		JSR	COUT	
FF37: 20 ED FD		JSR	COUT	
FF3A: A9 87	BELL	LDA	#S87	OUTPUT BELL AND RETURN
FF3C: 4C ED FD		JAP	COUT	
FF3F: A5 46	RESTORE	LDA	STATUS	RESTORE 6502 PFG CONTENTS
FF41: 48		PHA		USED BY DEBUG SOFTWARE
FF42: A5 45		LDA	ACC	
FF44: A6 46	RESTRI	LDX	XREG	
FF46: A4 47		LDY	YREG	
FF48: 28		PLP		
FF49: 60		RTS		
FF4A: 85 45	SAVE	STA	ACC	SAVE 6502 REG CONTENTS
FF4C: 86 46	SAV1	STX	XREG	
FF4E: 84 47		STY	YREG	
FF50: 08		PHP		
FF51: 68		PLA		
FF52: 85 48		STA	STATUS	
FF54: BA		TSX		
FF55: 86 49		STX	SPNT	
FF57: D8		CLD		
FF58: 60		RTS		
FF59: 20 84 FE	RESET	JSR	SETNORM	SFU SCREEN MODE
FF5C: 20 2F FB		JSR	INIT	AND INIT KBD/SCREEN
FF5F: 20 93 FE		JSR	SETVID	AS I/O DEV'S
FF62: 20 89 FE		JSR	SETKBD	
FF65: D8	MON	CLD		MUST SET HEX MODE!
FF66: 20 3A FF		JSR	BELL	
FF69: A9 AA	MONZ	LDA	#SAA	'*' PROMPT FOR MON
FF6B: 85 33		STA	PROMPT	
FF6D: 20 67 FD		JSR	GETLNZ	READ A LINE

FF70: 20 C7 FF		JSR ZMODE	CLEAR MON MODE, SCAN IDX
FF73: 20 A7 FF	NXTITM	JSF GETNUM	GET ITEM, NON-HEX
FF76: 84 34		STY YSAV	CHAR IN A-REG
FF78: A0 17		LDY #S17	X-REG=0 IF NO HEX INPUT
FF7A: 88	CHRSRCH	DEY	
FF7B: 30 E8		RMI MON	NOT FOUND, GO TO MON
FF7D: 09 CC FF		CMF CRTBL,Y	FIND CMND CHAR IN TFL
FF80: D0 F8		BNE CHRSPCH	
FF82: 20 BE FF		JSR TOSUB	FOUND, CALL CORRESPONDING
FF85: A4 34		LDY YSAV	SUPROUTINE
FF87: 4C 73 FF		JMP NXTITM	
FF8A: A2 03	DIC	LDX #S03	
FF8C: 0A		ASL A	
FF8D: 0A		ASL A	GOT HEX DIC,
FF8E: 0A		ASL A	SHIFT INTO A2
FF8F: 0A		ASL A	
FF90: 0A	NXTBIT	ASL A	
FF91: 26 3E		ROL A2L	
FF93: 26 3F		ROL A2H	
FF95: CA		DEX	LEAVE X=\$FF IF DIC
FF96: 10 F8		BPL NXTBIT	
FF98: A5 31	NXTES	LDA MODE	
FF9A: D0 06		BNE NXTES2	IF MODE IS ZERO
FF9C: B5 3F		LDA A2H,X	THEN COPY A2 TO
FF9E: 95 3D		STA A1H,X	A1 AND A3
FFA0: 95 41		STA A3H,X	
FFA2: E8	NXTES2	INX	
FFA3: F0 F3		BEQ NXTES	
FFA5: D0 06		BNE NXTCHR	
FFA7: A2 00	GETNUM	LDX #S0C	CLEAR A2
FFA9: 86 3E		STX A2L	
FFAB: 86 3F		STX A2H	
FFAD: B9 00 02	NXTCHR	LDA IN,Y	GET CHAR
FFB0: C8		INX	
FFB1: 49 30		FOR #S80	
FFB3: C9 0A		CMF #S0A	
FFB5: 90 03		BCC GIG	IF HEX DIG, THEN
FFB7: 69 86		ADC #S88	
FFB9: C9 FA		CMF #SFA	
FFBB: B0 CD		BCC DIC	
FFBD: 60		RTS	
FFBE: A9 FE	TOSUB	LDA #CO/256	PUSH HIGH-ORDER
FFC0: 48		PHA	SUBR ADR ON STK
FFC1: B9 E3 FF		LDA CRTBL,Y	PUSH LO ORDER
FFC4: 48		PHA	SUBR ADR ON STK
FFC5: A5 31		LDA MODE	
FFC7: A0 00	ZMODE	LDY #S0C	CLP MODE, OLD MODE
FFC9: 44 31		STY MODE	TO A-REG
FFCB: 60		RTS	GO TO SUBR VIA RTS
FFCC: BC	CHRTPL	DFB \$3C	F("CTRL-C")
FFCD: B2		DFB \$32	F("CTRL-Y")
FFCE: BE		DFB \$3E	F("CTRL-E")
FFCF: LD		DFB \$F0	F("")
FFD0: EF		DFB \$FF	F("")
FFD1: C4		DFB \$C4	F("CTRL-K")
FFD2: EC		DFB \$FC	F("S")
FFD3: A9		DFB \$A9	F("CTRL-P")
FFD4: B8		DFB \$B8	F("CTRL-R")
FFD5: A6		DFB \$A6	F("-")
FFD6: A4		DFB \$A4	F("+")
FFD7: 06		DFB \$06	F("M") (P=EX-OP \$B0+\$B9)
FFD8: 95		DFB \$95	F("<")
FFD9: 07		DFB \$07	F("N")
FFDA: 02		DFB \$02	F("I")
FFDB: 05		DFB \$05	F("L")
FFDC: F0		DFB \$F0	F(">")
FFDD: 00		DFB \$00	F("G")
FFDE: EB		DFB \$FB	F("R")
FFDF: 93		DFB \$93	F(":")
FFE0: A7		DFB \$A7	F(".")
FFE1: C6		DFB \$C6	F("CR")
FFE2: 99		DFB \$99	F(BLANK)
FFE3: B2	SUPTPL	DFB #BASCONT-1	
FFE4: C9		DFB #USR-1	
FFE5: BE		DFB #RFGZ-1	
FFE6: C1		DFB #TRACE-1	
FFEL7: 35		DFB #VPY-1	
FFE8: 8C		DFB #INPR-1	
FFE9: C3		DFB #STEPZ-1	
FFFA: 96		DFB #OUTPR-1	
FFEB: AF		DFB #XPASIC-1	
FFEC: 17		DFB #SETMODE-1	
FFED: 17		DFB #SETMODE-1	
FFEE: 2B		DFB #MOVE-1	
FFEF: 1F		DFB #LT-1	

```

*****
*
*   APPLE-II PSEUDO
* MACHINE INTERPRETER
*
*   COPYRIGHT 1977
* APPLE COMPUTER INC
*
* ALL RIGHTS RESERVED
*
*   S. WOZNIAK
*
*****

```

TITLE "SWEET16 INTERPRETER"

```

ROL      EPZ  $0
R0H      EPZ  $1
R14H     EPZ  $1D
R15L     EPZ  $1E
R15H     EPZ  $1F
S16PAG   EQU  SF7
SAVE     EQU  SFF4A
RESTORE  EQU  SFF3F
          ORG  SF689

F689: 20 4A FF SW16 JSR SAVE PRESERVE 6502 REG CONTENTS
F68C: 68          PLA
F68D: 85 1E      STA R15L INIT SWEET16 PC
F68F: 68          PLA FROM RETURN
F690: 85 1F      STA F15H ADDRESS
F692: 20 98 F6 SW16 JSP SW16C INTERPRET AND EXECUTE
F695: 4C 92 F6 SW16 JMP SW16E ONE SWEET16 INSTR.
F698: E6 1E      INC R15L
F69A: D0 02      BNE SW16D INCP SWEET16 PC FOR FETCH
F69C: E6 1F      INC F15H
F69E: A9 F7 SW16D LDA #S16PAG
F6A0: 48          PHA PUSH ON STACK FOR PTS
F6A1: A0 00      LDY #S0
F6A3: E1 1E      LDA (R15L),Y FETCH INSTR
F6A5: 29 0F      AND #SF MASK REG SPECIFICATION
F6A7: 0A          ASL A DOUBLE FOR 2-BYTE REGISTERS
F6A8: AA          TAX TO X-REG FOR INDEXING
F6A9: 4A          LSR A
F6AA: 51 1E      EOR (R15L),Y NOW HAVE OPCODE
F6AC: F0 0E      BEQ TOPF IF ZERO THEN NON-REG OP
F6AE: 86 1D      STX R14H INDICATE 'PRIOR RESULT REG'
F6B0: 4A          LSR A
F6B1: 4A          LSR A OPCODE*2 TO LSP'S
F6B2: 4A          LSR A
F6B3: A8          TAY TO Y-REG FOR INDEXING
F6B4: E9 F1 F6 LDA CPTBL-2,Y LOW-ORDER ADR BYTE
F6B7: 48          PHA ONTO STACK
F6B8: 60          RTS GOTO REG-OP ROUTINE
F6B9: E6 1E TO3R INC R15L
F6BB: D0 02      BNE TOPR2 INCR PC
F6BD: E6 1F      INC R15H
F6BF: ED E4 F6 TOPR2 LDA SETPL,X LOW-ORDER ADR BYTE
F6C2: 48          PHA ONTO STACK FOR NON-REG OP
F6C3: A5 1D      LDA R14H 'PRIOR RESULT REG' INDEX
F6C5: 4A          LSR A PREPARE CARRY FOR BC, BNC.
F6C6: 60          RTS GOTO NON-REG OP ROUTINE
F6C7: 68          PLR POP RETURN ADDRESS
F6C8: 68          PLA
F6C9: 20 3F FF JSR RESTORE RESTORE 6502 REG CONTENTS
F6CC: 6C 1E 00 JMP (R15L) RETURN TO 6502 CODE VIA PC
F6CF: E1 1F SETZ LDA (R15L),Y HIGH-ORDER BYTE OF CONSTANT

```


F6D1: 95 01		STA	ROH,X	
F6D3: 88		DEFY		
F6D4: B1 1E		LDA	(R15L),Y	LOW-ORDER BYTE OF CONSTANT
F6D6: 95 00		STA	ROL,X	
F6D8: 98		TYA		Y-REG CONTAINS 1
F6D9: 38		SEC		
F6DA: 65 1E		ADC	R15L	ADD 2 TO PC
F6DC: 85 1E		STA	R15L	
F6DE: 90 02		BCC	SET2	
F6E0: E6 1F		INC	R15H	
F6E2: 60	SFT2	RTS		
F6E3: 02	OPTBL	DEF	SLT-1	(1X)
F6E4: F9	PRIBL	DEF	RIN-1	(0)
F6E5: 04		DEF	LD-1	(2X)
F6E6: 9D		DEF	CR-1	(1)
F6E7: 0D		DEF	ST-1	(3X)
F6E8: 9E		DEF	RMC-1	(2)
F6E9: 25		DEF	IDAT-1	(4X)
F6EA: AF		DEF	RC-1	(3)
F6EB: 16		DEF	STAT-1	(5X)
F6EC: B2		DEF	BP-1	(4)
F6ED: 47		DEF	LDDAT-1	(6X)
F6EE: B9		DEF	SM-1	(5)
F6EF: 51		DEF	STDAT-1	(7X)
F6F0: C0		DEF	RZ-1	(6)
F6F1: 2F		DEF	POP-1	(8X)
F6F2: C9		DEF	RNZ-1	(7)
F6F3: 5B		DEF	STPAT-1	(9X)
F6F4: D2		DEF	RM1-1	(8)
F6F5: 85		DEF	ADD-1	(AX)
F6F6: DD		DEF	BNM1-1	(9)
F6F7: 6E		DEF	SU3-1	(8X)
F6F8: 05		DEF	RK-1	(A)
F6F9: 33		DEF	PCPD-1	(CX)
F6FA: E8		DEF	RS-1	(R)
F6FB: 70		DEF	CPR-1	(DX)
F6FC: 93		DEF	BS-1	(C)
F6FD: 1E		DEF	INR-1	(EX)
F6FE: E7		DEF	NUL-1	(D)
F6FF: 65		DEF	DCR-1	(FX)
F700: E7		DEF	NUL-1	(E)
F701: E7		DEF	NUL-1	(UNUSED)
F702: E7		DEF	NUL-1	(F)
F703: 10 CA	SET	BPL	SETZ	ALWAYS TAKEN
F705: B5 00	LD	LDA	ROL,X	
	BK	EQU	*-1	
F707: 85 00		STA	ROL	
F709: B5 01		LDA	ROH,X	MOVE RX TO R0
F70B: 85 01		STA	ROH	
F70D: 60		RTS		
F70E: A5 00	ST	LDA	ROL	
F710: 95 00		STA	ROL,X	MOVE R0 TO RX
F712: A5 01		LDA	ROH	
F714: 95 01		STA	ROH,X	
F716: 60		RTS		
F717: A5 00	STAT	LDA	ROL	
F719: 81 00	STAT2	STA	(ROL,X)	STORE BYTE INDIRECT
F71B: A0 00		LDY	#S0	
F71D: 84 1D	STAT3	STY	R14H	INDICATE R0 IS RESULT REG
F71F: F6 00	INR	INC	ROL,X	
F721: D0 02		BNE	INR2	INCR RX
F723: F6 01		INC	ROH,X	
F725: 60	INR2	RTS		
F726: A1 00	LDA	LDA	(ROL,X)	LOAD INDIRECT (RX)
F728: 85 00		STA	ROL	TO R0
F72A: A0 00		LDY	#S0	
F72C: 84 01		STY	ROH	ZERO HIGH-ORDER R0 BYTE
F72E: F0 ED		BEO	STAT3	ALWAYS TAKEN
F730: A0 00	POP	LDY	#S0	HIGH ORDER BYTE = 0
F732: F0 06		BEO	POP2	ALWAYS TAKEN
F734: 20 66 F7	POPD	JSR	DCR	DECR RX
F737: A1 00		LDA	(ROL,X)	POP HIGH-ORDER BYTE @RX
F739: A8		TAY		SAVE IN Y-REG
F73A: 20 66 F7	POP2	JSR	DCP	DECR RX
F73D: A1 00		LDA	(ROL,X)	LOW-ORDER BYTE
F73F: 85 00		STA	ROL	TO R0
F741: 84 01		STY	ROH	
F743: A0 00	POP3	LDY	#S0	INDICATE R0 AS LAST RSLT REG
F745: 84 1D		STY	R14H	
F747: 60		RTS		
F748: 20 26 F7	LDDAT	JSR	LDA	LOW-ORDER BYTE TO R0, INCR RX
F74B: A1 00		LDA	(ROL,X)	HIGH-ORDER BYTE TO R0
F74D: 85 01		STA	ROH	
F74F: 4C 1F F7		JMP	INR	INCR RX
F752: 20 17 F7	STDAT	JSR	STAT	STORE INDIRECT LOW-ORDER

F755: A5 01		LDA R0H	RYTF AND INCR RX. THEN
F757: 81 00		STA (R0L,X)	STORE HIGH-ORDER BYTE.
F759: 4C 1F F7		JMP INR	INCR RX AND RETURN
F75C: 20 66 F7 STPAT		JSR DCF	DECR RX
F75F: A5 00		LDA R0L	
F761: 81 00		STA (R0L,X)	STORE R0 LOW BYTE 0RX
F763: 4C 43 F7		JMP POP3	INDICATE R0 AS LAST RSLT REG
F766: B5 00	DCR	LDA R0L,X	
F768: D0 02		SNE DCR2	DECR PX
F76A: D6 01		DEC R0H,X	
F76C: D6 00	DCR2	DEC R0L,X	
F76E: 60		RTS	
F76F: A0 00	SU3	LDY #S0	RESULT TO R0
F771: 38	CPR	SEC	NOTE Y-REG = 13*2 FOR CPR
F772: A5 00		LDA R0L	
F774: F5 00		SBC R0L,X	
F776: 99 00 00		STA R0L,Y	R0-RX TO RY
F779: A5 01		LDA R0H	
F77B: F5 01		SBC R0H,X	
F77D: 99 01 00 SUB2		STA R0H,Y	
F780: 98		TYA	LAST RESULT REG*2
F781: 69 00		ADC #S0	CARRY TO LSR
F783: 85 1D		STA R14H	
F785: 60		RTS	
F786: A5 00	ADD	LDA R0L	
F788: 75 00		ADC R0L,X	
F78A: 85 00		STA R0L	R0+RX TO R0
F78C: A5 01		LDA R0H	
F78E: 75 01		ADC R0H,X	
F790: A0 00		LDY #S0	R0 FOR RESULT
F792: F0 E9		BEO SUB2	FINISH ADD
F794: A5 1E	BS	LDA R15L	NOTE X-REG IS 12*2!
F796: 20 19 F7		JSR STAT2	PUSH LOW PC BYTE VIA R12
F799: A5 1F		LDA R15H	
F79B: 20 19 F7		JSR STAT2	PUSH HIGH-ORDER PC BYTE
F79E: 18	BR	CLC	
F79F: B0 0E	SNC	ECS BNC2	NO CARRY TEST
F7A1: B1 1E	BR1	LDA (R15I),Y	DISPLACEMENT BYTE
F7A3: 10 01		SPL BR2	
F7A5: 88		DEY	
F7A6: 65 1E	BR2	ADC R15L	ADD TO PC
F7A8: 85 1E		STA R15L	
F7AA: 98		TYA	
F7AB: 65 1F		ADC R15H	
F7AD: 85 1F		STA R15H	
F7AF: 60	SNC2	RTS	
F7B0: B0 EC	BC	BCS BR	
F7B2: 60		RTS	
F7B3: 0A	BP	ASL A	DOUBLE RESULT-REG INDEX
F7B4: AA		TAX	TO X-REG FOR INDEXING
F7B5: B5 01		LDA R0H,X	TEST FOR PLUS
F7B7: 10 E8		BPL BP1	BRANCH IF SO
F7B9: 60		RTS	
F7BA: 0A	BR	ASL A	DOUBLE RESULT-REG INDEX
F7BB: AA		TAX	
F7BC: B5 01		LDA R0H,X	TEST FOR MINUS
F7BE: 30 E1		BMI BP1	
F7C0: 60		RTS	
F7C1: 0A	BR	ASL A	DOUBLE RESULT-REG INDEX
F7C2: AA		TAX	
F7C3: B5 00		LDA R0L,X	TEST FOR ZERO
F7C5: 15 01		ORA R0H,X	(BOTH BYTES)
F7C7: F0 D8		BEO BP1	BRANCH IF SO
F7C9: 60		RTS	
F7CA: 0A	BRZ	ASL A	DOUBLE RESULT-REG INDEX
F7CB: AA		TAX	
F7CC: B5 00		LDA R0L,X	TEST FOR NONZERO
F7CE: 15 01		ORA R0H,X	(BOTH BYTES)
F7D0: D0 CF		BNE BR1	BRANCH IF SO
F7D2: 60		RTS	
F7D3: 0A	BR1	ASL A	DOUBLE RESULT-REG INDEX
F7D4: AA		TAX	
F7D5: B5 00		LDA R0L,X	CHECK BOTH BYTES
F7D7: 35 01		AND R0H,X	FOR \$FF (MINUS 1)
F7D9: 49 FF		EOR #\$FF	
F7DB: F0 C4		BEQ BR1	BRANCH IF SO
F7DD: 60		RTS	
F7DE: 0A	BNM1	ASL A	DOUBLE RESULT-REG INDEX
F7DF: AA		TAX	
F7E0: B5 00		LDA R0L,X	
F7E2: 35 01		AND R0H,X	CHECK BOTH BYTES FOR NO \$FF
F7E4: 49 FF		EOR #\$FF	
F7E6: D0 B9		BNE BP1	BRANCH IF NOT MINUS 1
F7E8: 60	NUL	RTS	
F7E9: A2 18	RS	LDX #\$18	12*2 FOR R12 AS STK POINTER

F7E8:	20 66 F7	JSR	DCR	DECR STACK POINTER
F7EE:	A1 00	LDA	(R0L,X)	POP HIGH RETURN ADR TO PC
F7F0:	85 1F	STA	R15H	
F7F2:	20 66 F7	JSR	DCR	SAME FOR LOW-ORDER BYTE
F7F5:	A1 00	LDA	(R0L,X)	
F7F7:	85 1E	STA	R15L	
F7F9:	60	RTS		
F7FA:	4C C7 F6 RTN	JMP	RTNZ	

HOOFDSTUK 9 DISK DRIVES

1. HET GEBRUIK VAN DISK DRIVES

De disk drive eenheden van de APPLE-II computer vormen een welkome aanwinst voor de professionele gebruiker. De disk drive verschaft in weinig ogenblikken toegang tot grote hoeveelheden magnetisch vastgelegde informatie. Het z.g. DISK-II systeem van de APPLE kan ruim 120.000 tekens opslaan op kleine, flexibele, magnetische diskettes. Deze diskettes zijn verpakt in een stijve, vierkante enveloppe van ca. 12 cm². De computerinformatie wordt er -evenals bij een cassettebandje- door een elektronische lees/schrijfkop op overgebracht. De computer bestuurt deze kop en kan hem boven elke gewenste positie op de diskette brengen. Hierdoor wordt de diskette min of meer willekeurig toegankelijk. Er is sprake van "Random Access".

Om de disk drive te besturen, is er speciale programmatuur geschreven. Voor een deel is de besturingslogica in de disk drive zelf aanwezig. Om de disk drive evenwel met de computer te verbinden, is er een "tussenschakel" nodig: de z.g. Interface. De disk interface bevat eveneens besturingslogica. Tenslotte wordt door de computer nog een lees/schrijfprogramma op de diskettes geschreven, zodat ze -na te zijn "geïntialiseerd"- door de disketttestations kunnen worden "gelezen".

Dit geheel van programmatuur vormt het DISK OPERATING SYSTEM: D.O.S. Het D.O.S. bestuurt de gegevensoverdracht tussen disketttestations en computer.



TRACKS & SECTOREN

Om het mogelijk te maken een bepaald gegeven van de diskette te lezen, verdeelt het APPLE DOS de kleine diskette in 35 concentrische ringen. Deze informatieringen zijn genummerd van 0 t/m 34. Men noemt de ringen "tracks". Track 0 vormt de buitenste ring van de diskette, track 17 de middelste en track 34 tenslotte de binnenste ring. In zekere zin kun je de tracks vergelijken met de groeven van een gramfoonplaat. Echter zijn de tracks dus geen informatie-spiraal, zoals de groef van een gramfoonplaat: het zijn ringen, die in elkaar liggen. Evenals dit het geval is bij een platenspeler draait de diskette met een konstante snelheid rond, terwijl gegevens van en naar zijn oppervlak worden geschreven.

Een "stappenmotor" in de disk drive zorgt er nu voor, dat de opname/weergavekop gelijkmatig boven de diskette wordt heen en weer geschoven. Om de kop 1 track te laten verschuiven zijn 2 fasen van de stappenmotor vereist. Dus in principe zou de diskette tweemaal 35 tracks kunnen bevatten, ware het niet, dat de lees/schrijfkop een onvoldoende gevoeligheid heeft voor dergelijke kleine stappen. Hij zou niet meer herkennen boven welke track hij zich bevond.

Een sector is een onderverdeling van de track. Het DOS schrijft meestal sector voor sector, en leest zo eveneens. Hierdoor wordt voorkomen, dat een te groot deel van het geheugen gebruikt moet worden als

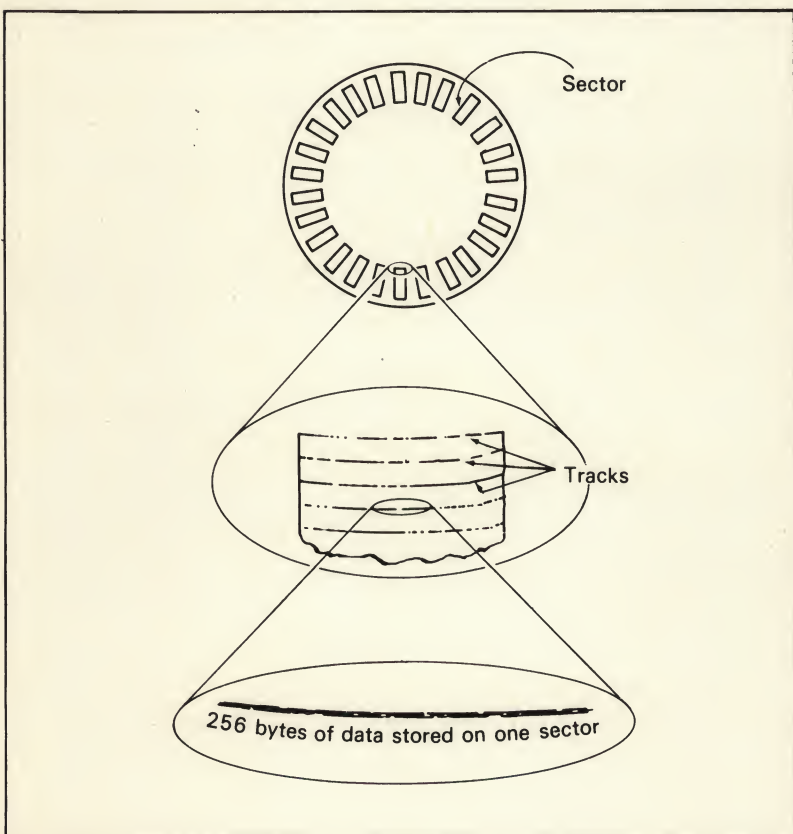
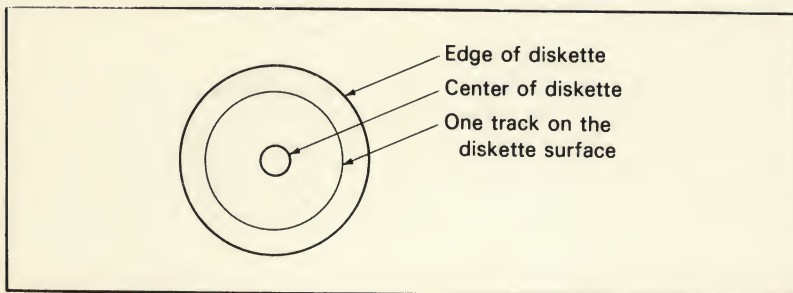
"voorraadschuur" voor over te zetten gegevens. Een "voorraadschuur" met gegevens noemt men een buffer. Het APPLE DOS 3.3 kent aan een track 16 sectoren toe. Het vroegere DOS 3.2 (.1) maar 13 sectoren. Elke sector bevat 256 bytes informatie. Je kunt nu dus uitrekenen welke gegevenscapaciteit een mini-diskette heeft.

De kracht van het DISK OPERATING SYSTEM schuilt erin, dat het d.m.v. een soort inhoudsopgave, die het zelfstandig bijhoudt, elke gezochte byte kan vinden. Wanneer je immers de track en de sector weet, waar zich die byte moet ophouden, is het zoeken verder weinig opwindends!

Toch is er nog een bijzonderheid te vermelden. In de diskette zit afgezien van het grote centreringsgat een kleine opening. Men noemt dit gat het "Indexgat". Wanneer de diskette wordt rondgedraaid, valt er door dit kleine gat een lichtbundeltje op een sensor, die de computer dan precies kan laten weten, waar zich de eerste sector bevindt. Op grond van de waarneming door die sensor kan de computer vervolgens alle overige sectoren berekenen en vinden.

Op de verpakking van dit type diskettes staat, dat zij "soft-sectored" zijn. Dit doet je direkt vermoeden, dat er dus ook "hard-sectored" diskettes bestaan. Dat is juist!

De hard-sectored diskettes zijn voorzien van een reeks gaatjes, die ieder het begin van een sector aanduiden. De APPLE DISK II kan zoveel soft- als hard-sectored diskettes verwerken.



3. HET HANTEREN VAN DE DISKETTE

Het behoeft geen betoog, dat je voorzichtig moet zijn met de diskettes, of floppy disks. Hoewel zij verpakt zijn in een kunststof hoesje, dat op zijn beurt in een kartonnen enveloppe past, blijven het kwetsbare informatiedragers. Er gelden een aantal regels voor het gebruik van floppy disks, die we hier ter overweging geven:

- raak nimmer het open lees/schrijfvenster aan.
diskettes verdragen geen vingerafdrukken.
- houdt de diskette zoveel mogelijk stofvrij.
laat diskettes niet rondslingeren/berg ze op!
- doe de diskette voorzichtig in de disk drive.
voorkom kreuken en vouwen.
- houdt de diskette uit een magnetische omgeving.
leg geen diskette op de TV-monitor!
- houdt de diskette uit de volle zon.
boven 40 graden C. legt hij het loodje.

Als je deze eenvoudige en logische regels volgt, kun je lang genoeg beleven van de diskette. Diskettes kun je bewaren in de kleine omslagdozen, waarin ze worden geleverd, maar ook in fraaie plastic opbergdozen. Bekijk ze eens bij je dichtstbijzijnde computershop.

Over de levensduur van diskettes zijn de deskundigen het niet eens. Wanneer moet je een diskette gaan vervangen? Ervaren programmeurs doen dit, wanneer er plotseling moeilijkheden gaan ontstaan met het "aanspreken" van de diskette. "Hij start niet meer goed op," noemen zij dit wel. Wanneer dat niet wordt veroorzaakt door vervuiling van de DISK INTERFACEKAART in de computer zelf, is de diskette aan zijn pensioen toe. Hij is dan nog goed genoeg voor leuke spelletjes of probeersels, maar niet meer betrouwbaar voor belangrijke gegevens-bestanden.

4. WELK DISK-SYSTEEM IS VOOR MIJ GESCHIKT ?

Welk disksysteem uiteindelijk het meest in aanmerking komt bij de APPLE-II is in belangrijke mate afhankelijk van de toepassing. Gewoonlijk zal het compacte APPLE-DISK II Systeem voldoende zijn. Het DISK II Systeem heeft een bescheiden capaciteit met vooralsnog niet meer dan 120.000 bytes informatie per diskette. Het is aannemelijk, dat deze waarde binnenkort verdubbeld kan worden. Per DISK INTERFACE kaart kunnen 2 disk drives worden aangestuurd, hetgeen voor meer professioneel gebruik een noodzaak is. Een disk drive is hiervoor ontoereikend.

Wie evenwel met grote hoeveelheden gegevens moet werken, zal niet voldoende hebben aan 2 kleine disk drives. Meer in aanmerking komen dan de grotere, 8-inch disksystemen, waarvan er maar weinig echt aansluiten op het uitstekende APPLE DOS. Een zeer betrouwbaar en volmaakt compatibel systeem is b.v. het voordelige "MULTIBYTE"-diskstation. De "MULTIBYTE" diskdrives zijn de grote broers van de 5 1/4-inch DISK II ministations van APPLE. Deze grote broers zijn bestemd voor zelfstandigen, ondernemers, mensen in vrije beroepen en bedrijfsafdelingen, die een professioneel gebruik willen maken van de APPLE computer. De capaciteit van de 8-inch maxi-floppies loopt van 0.6 tot 4.8 Megabyte. Een Megabyte is 1 miljoen bytes. Wie hieraan nog niet voldoende heeft, kan een beroep doen op enkele typen vaste schijfengeheugens. Het gaat hierbij om disk drives, waarin een of meer vaste magneetschijven werden aangebracht. Door hun speciale bouw kunnen deze "hard-disks" van 1.2 tot wel 100 Megabyte bevatten. Maar het is zeer de vraag, of dit type schijfengeheugens onder alle omstandigheden zal blijven voldoen. Wie loopt er immers graag in een veel te groot pak rond ? En wie zou het zonder copieën willen stellen van zijn grote databestand ? De professionele gebruiker doet er zeker goed aan zich vooraf door een deskundig adviesbureau te laten inlichten.

Af te raden is het, om voor zuiver professioneel gebruik gebruik te maken van allerlei "koopjes", die met name vaak op de hobbemarkt zijn te vinden. Net zomin je je bedrijfsgegevens op een WC-papiertje schrijft en bewaart, kun je belangrijke computergegevens bewaren op een floppy-disk, die door de fabriek bijna werd afgekeurd en "goedkoop" op de markt werd gedumpt! Hetzelfde geldt voor diskette-stations. Alle waar is naar zijn geld ..., zelfs in computerland.

5. VAN PR#TOT CATALOG

Bij de aanschaf van je diskette-station(s) kreeg je een "DOS MANUAL" uitgereikt, alsmede een of meer diskettes. Daaronder bevindt zich zonder twijfel een z.g. MASTER diskette.

Alvorens met de diskettes te kunnen werken, zullen wij de diskdrive(s) aan moeten sluiten op de APPLE COMPUTER. Daartoe nemen we het deksel van de APPLE II voorzichtig af. Schakelen de stroom uit en steken de DISK INTERFACEKAART, waaraan de platte stationskabel vast moet zitten, in Slot 6. Hierna sluiten wij de computer en schakelen we de stroom weer in. Er kunnen nu 2 dingen gebeuren:

1. het lichtje op de diskdrive gaat aan
2. je ziet alleen een rommeltje op het beeldscherm met het * teken als prompt.

In het eerste geval zorgt de Autostart ROM van de APPLE II ervoor, dat direkt na het inschakelen van de computer informatie wordt gevraagd van het disktestation. In het tweede geval zullen we met de hand enkele voorbereidende stappen moeten zetten.

Om in het eerste en thans meest voorkomende geval de diskdrive tot stilstand te brengen, is een druk op de RESET-toets nodig. Eventueel CTRL-RESET. Doe nu voorzichtig de klep van het diskstation open en schuif de master diskette zachtjes naar binnen. Weest voorzichtig voor deze vitale systeem-master: je zult hem nog vaak nodig hebben. Sluit de klep van de diskdrive. Tik nu vanuit Applesoft het volgende commando:

PR#6 (RETURN)

De diskdrive moet nu werken. Het rode "IN USE"-lichtje gaat aan. En na enig geratel en geschuifel verschijnt een tekst op het beeldscherm, die kennelijk van de masterdiskette afkomstig is. Gefeliciteerd!

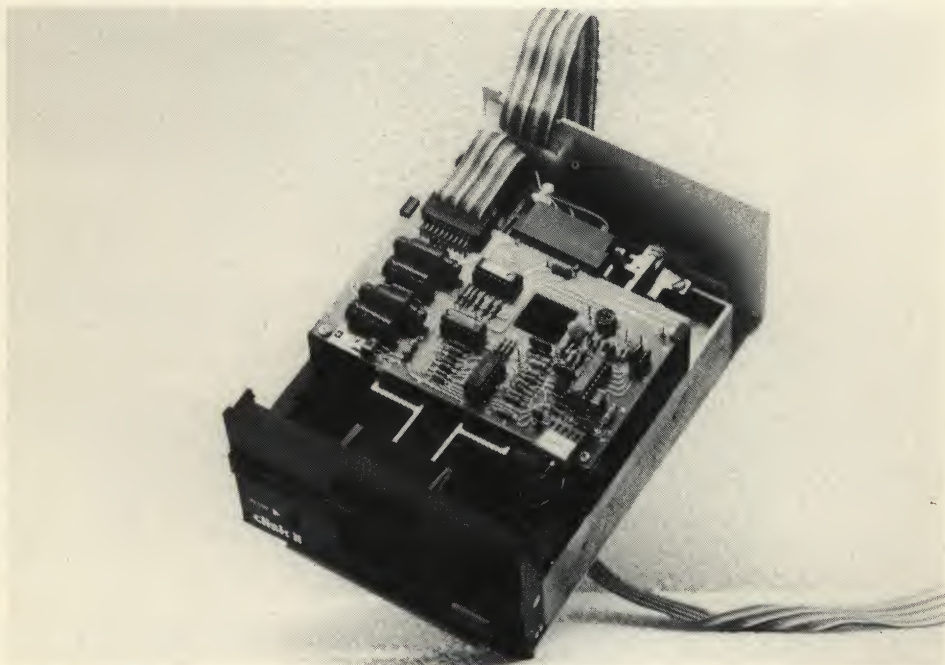
Maar wat moet je doen, om te weten, wat er verder op de schijf aanwezig is? Een masterdiskette staat vol met nuttige programma's, die je later veel zult willen gebruiken. Tik daarom het volgende commando:

CATALOG (RETURN)

Weer die enge ratel geluiden, maar ... kijk eens aan ! Dat staat dus op je master-diskette.

Het CATALOG-commando geeft voor elk programma op de schijf een overzicht van het programma-type, het aantal sectoren dat het programma inneemt, of het programma tegen ongewild overschrijven is beveiligd (ge-locked -*), en de naam van het programma.

Is de schijf zo vol met programma's, dat er meer dan 20 regels voor het afbeelden van de CATALOG nodig zijn, dan stopt de computer even en wacht tot je op het keyboard tikt alvorens de rest van de CATALOG te laten zien. Wie gelijk met een 8-inch systeem is gaan werken, dat APPLE DOS compatibel is, zal op dezelfde wijze kunnen werken, doch meestal zal de DISK INTERFACE in Slot 7 resideren, wat tot gevolg heeft, dat het opstart-commando PRH/ wordt. We gaan er verder vanuit, dat met de APPLE DISK II stations wordt gewerkt.



6.DIREKTE D.O.S.-COMMANDO'S

Bij de aflevering van de APPLE DISK II stations wordt het z.g. DOS MANUAL uitgereikt. Het is zeer aan te bevelen, dit uitstekende handboek door te lezen. Wie gebruik maakt van andere, maar APPLE compatibele diskettestations (b.v. de MultiBytes), doet er goed aan de handleiding bij het systeem voorafgaand aan verder gebruik door te nemen. "Waarom deze overbodige regels?," zul je vragen. Het is helaas maar al te vaak voorgekomen, dat gebruiksmoeilijkheden ontstonden door onvoldoende inzicht in de eigenaardigheden van het disksubstelsysteem. Hoe vaak er niet belangrijke data-bestanden zijn verloren gegaan door onjuist disk-gebruik, kan men zich nauwelijks voorstellen!

De hierna volgende diskcommando's geven een summier overzicht van de mogelijkheden van het APPLE DOS.

LOAD <filenaam>

Het LOAD commando leest een programma van de diskette en plaatst het in het werkgeheugen. Indien de opgegeven programmanaam niet op de diskette voorkomt, volgt de foutmelding FILE NOT FOUND.

Als het aangeroepen programma een Binary file is, of een Text file, dan volgt de foutmelding FILE TYPE MISMATCH. LOAD wist het programma, dat nog in het werkgeheugen resideert en schrijft het aangeroepen programma daarin over. Hierna kan het programma worden gerund, of gemodificeerd.

RUN <filenaam>

Om het gebruiksgemak op te voeren, werd een speciaal DOS RUN commando opgenomen. Het is niet noodzakelijk eerst een programma te loaden en het daarna te runnen. Je kunt direkt een programma op de diskette gaan runnen. Het commando RUN COPYA brengt direkt het APPLE copieerprogramma in werking, dat wij nodig zullen hebben voor het maken van onze eerste back-up copie. Enkel van de systeem-master diskette. Doe een lege diskette in je tweede diskettestation, of -indien je dit nog niet hebt- houdt de lege diskette bij de hand. Volg de aanwijzingen van het COPYA programma op en binnen enkele ogenblikken is de copie van de masterdiskette gereed voor gebruik.

INIT <filenaam>,(<slot>,<drive>,<volume>)

Voor je een diskette kunt gebruiken, dient hij z.g. geïnitieerd te zijn. Mocht je nu denken, dat achter dit moeilijke woord een oeroude inwijdingsrite schuil gaat, dan heb je het eigenlijk niet eens mis. Kijk, een diskette is bij aflevering meestal een onbeschreven blad. Hoe kan het DOS daar nu zijn weg op vinden ? Het initialiserings-proces zorgt hiervoor en tijdens dit proces wordt de gehele schijf schoongeveegd voor nieuw gebruik. De tracks en sectoren worden gemarkeerd. Door middel van frequentiemodulatie worden synchronisatiebits

op de schijf geschreven, waardoor het mogelijk wordt dat de computer later precies de goede track en sector met je programma kan vinden.

Zoals je misschien bij het copieren van je masterdiskette hebt opgemerkt, initialiseerde het COPYA-programma de nieuwe schijf eerst, alvorens de masterdiskette te copieren. Het programma noemde dit "formatting". Bij grote 8-inch diskettes gaat het "formatteren" zelfs aan het initialiseren vooraf!

Het INIT commando is van veel betekenis. Het maakt je diskette gereed voor gebruik. Geef je het INIT commando, dan gebruikt de computer het in het werkgeheugen aanwezige programma als toekomstig en praktisch onvervangbaar "begroetingsprogramma". Het is dus uitkijken geblazen met INIT ! Beter is, om een standaard gewoonte aan te leren voor het maken van een begroetingsprogramma. Hier volgt een voorbeeld:

```
] 10 REM INIT D.D. 05/12/81
   15 PRINT CHR$(4);"CATALOG"
   20 END
```

Doe nu een lege schijf in de diskdrive en tik INIT HELLO,D1 (of 2?),S6 (of ander Slot, je hoeft dit niet te specificeren),V001 (volume mag je weglaten, vervalt dan naar 254).

```
]INIT HELLO,D1,S6,V1.
```

Nu gaat het IN USE lichtje van de opgegeven drive branden en komen de bekende klikkende geluidjes uit de diskdrive. Ongeveer 30 seconden later heb je je eerste geïntialiseerde programmadiskette, die we verder zullen gebruiken.

Doe de geïntialiseerde diskette in drive 1 en tik PR#(slotnr.).

Begrijp je, wat er nu is gebeurd ?

Het is duidelijk, dat INIT alleen werkt, wanneer het DOS in de computer aanwezig is; dus na het opstarten met een geïntialiseerde diskette.

Wordt een Slot genoemd, waarin geen interface zit, dan "hangt" de computer tot RESET wordt gegeven.

Wordt in een programma een disk Volume nummer gegeven en stemt dat niet overeen met het Volumenummer van de ingeschoven diskette, dan volgt de melding VOLUME MISMATCH. Je moet een weldoordacht gebruik maken van deze mogelijkheid.

Wordt een Drive nummer gegeven, dat niet aanwezig is, dan kunnen er vreemde dingen gebeuren. Soms hangt de computer tot RESET wordt gegeven. Soms treedt een willekeurige drive in werking.

```
SAVE <filenaam>,<slot>,<drive>,<volume>
```


Met het SAVE commando is het mogelijk een programma op de diskette te bewaren. Probeer het volgende:

```
10 PRINT "DAAR GAAT IE DAN"
15 PRINT "MIJN EERSTE PROGRAMMA OP SCHIJF"
20 END
SAVE HUPLA
```

Lampje gaat aan; enig geschuifel en klaar. Tik nu RUN HUPLA en ...

LOCK <filenaam>

Met LOCK is het mogelijk een programma tegen ongewild wissen te beschermen, maar niet tegen initialiseren. Een ge-lock-te file is bij een CATALOG herkenbaar aan het * teken voor de programmaam.

UNLOCK <filenaam>

Het tegenovergestelde van LOCK. Het programma kan nu worden gewijzigd en teruggeschreven naar de diskette. Proberen we een gelockt programma na wijzigingen terug te schrijven, dan volgt FILE LOCKED.

RENAME <filenaam>

Met RENAME kunnen we een bestaand programma een andere naam geven. Het protocol luidt b.v.: RENAME HUPLA,HOEPLA (return). Oppassen met RENAME als je het INIT-programma hiermee wilt bewerken. De schijf zal bij het booten het oorspronkelijke begroetingsprogramma niet meer kunnen vinden en de boot-operatie stopt met de melding: FILE NOT FOUND

DELETE <filenaam>

DELETE wist een programma van de schijf en geeft de ingenomen tracks en sectoren vrij voor overschrijving door een nieuw op te nemen programma, zonder het oude vooralsnog te vernietigen. Alleen goed ingevoerde programmeurs kunnen een per ongeluk ge-delete programma nog terugroepen, zolang er verder maar niets op de schijf wordt ge-saved!

VERIFY <filenaam>

Een wat mysterieus commando, dat je in staat stelt bepaalde gegevens op de schijf te controleren. Indien een schijf is beschadigd kunnen gedeelten van de data zijn verminkt en klopt een verificatie van de data niet meer. De z.g. bit-checksum komt dan niet uit. De computer berekent dit voor je. Klopt het niet, dan volgt het tragische bericht: I/O ERROR. Het bestand is dan onbruikbaar gebleken.

7. DOS PROGRAMMA COMMANDO'S

lot zover hebben wij uitsluitend direkte DOS commando's besproken, die vanaf het keyboard werden ingegeven. Natuurlijk is het mogelijk DOS commando's te geven vanuit programma's. Een voorbeeld daarvan zag je reeds bij het INIT programma. Op dit thema gaan we hierna verder in.

Om het DOS vanuit een programma aan te roepen, moeten we een "disk-controle"-teken gebruiken. Heel toepasselijk wordt daarvoor bij het APPLE DOS het teken: CTRL-D gebruikt. CTRL-D, of ASCII-code 4, is het toverteken, dat het ons in staat stelt interaktief met het Disk Operating System te werken. Zoals gesteld zijn er 2 opties mogelijk, die beide hetzelfde effect hebben:

- a) CTRL-D intikken
- b) CHR\$(4) intikken

Nu zouden beide stuurcommando's in een programma wat uit de lucht komen vallen, daarom volgen ze direkt na een afzonderlijk PRINT statement. Dus: PRINT CHR\$(4), of PRINT"". Tussen de "" staat het onzichtbare CTRL-D teken. Omdat het Integer Basic van de oorspronkelijke APPLE II geen CHR\$-commando kent, zullen gebruikers van die taal terug moeten vallen op "", waartussen CTRL-D wordt getikt.

Een tamelijk ingevoerd gebruik is, dat men vooraan in een programma een string variabele benoemt, met de kwaliteit CTRL-D. Meestal gebruikt men hiervoor D\$. Een kort programma, dat een CATALOG teweegbrengt, ziet er daarom zo uit:

```
10 D$=CHR$(4)
15 PRINT D$;"CATALOG"
20 END
```

RUN dit programma en SAVE het, met een naam naar uw keuze. Misschien is dit wel een aardig INIT programma ?

Natuurlijk zijn de mogelijkheden veel groter en professioneler. Daarover nu het volgende. Het APPLE DOS laat toe, dat vanuit bepaalde programma's gegevens naar en van de schijf getransporteerd worden. Het is dus mogelijk een ledenadministratie, een boekhouding, een adresbestand, berekeningen en hun uitkomsten op de schijf op te slaan. Een computerbestand noemt men een File (spreek uit: faail). De APPLE II kent 2 soorten files. Ten eerste het sequentiele bestand; ten tweede het willekeurig toegankelijke bestand.

In een sequentieel bestand kan men slechts gegevens terugzoeken door bij het begin van het bestand met zoeken aan te vangen totdat men het gezochte onderwerp heeft gevonden. Een omslachtig gebeuren, dat niet altijd voordelen heeft.

In een "random access" bestand kan men op specificatie gegevens terugvinden, zonder dat de hele file doorzocht moet worden. Je kunt raden, dat deze werkwijze het meest wordt toegepast bij bedrijfsprogrammatuur.

A. HET SEQUENTIEEL BESTAND

De DOS commando's OPEN, READ/WRITE, CLOSE hebben betrekking op het werken met sequentiele files. Voordat men een sequentieel bestand kan aanleggen, zal het DOS een paar maatregelen moeten nemen. Zo zal het ruimte op de diskette moeten reserveren voor de file, en moeten controleren, of de genoemde file al aanwezig is. Voorts moet het DOS een buffer vrij maken voor het opvangen van gegevens, die ingelezen of weggeschreven worden. De truc werkt als volgt:

```
10 D$=CHR$(4) : REM CTRL-D WEET JE NOG
15 PRINT D$;"OPEN TRUC,S6,D1,V100"
```

Wil je Volume, noch Slotnummer, noch Drivenummer specificeren, dan mag dat! Werk je met meerdere drives, dan kan het nuttig zijn tenminste het drivenummer op te geven.

Ons programma OPENT nu een file, genaamd TRUC. Het frappante is, dat dit werkelijk te controleren is. Zelfs al hebben we nog geen gegevens naar de file geschreven, de file TRUC bestaat. Controleer dit door regel 20 op te nemen:

```
20 END
```

RUN dit programma en geef daarna een CATALOG commando. Zie je daar de file TRUC? Voor TRUC zie je 3 cijfers en daarvoor een T. De 3 cijfers geven het aantal door de file bezette sectoren aan, de T betekent, dat het hier om een Tekstfile gaat - een databestand.

Om gegevens naar de file te schrijven gebruiken we een WRITE statement. WRITE schrijft de gegevens, die erop volgen weg. De schrijfwijze voor ons programma wordt:

```
20 PRINT D$;"WRITE TRUC"
25 PRINT"DIT IS EEN GROTE TRUC!"
```

PAS UP! Hiermee is het programma niet af. Een UITERST belangrijk commando ontbreekt nog: CLOSE. Sluit de file af. Anders zou elk volgend PRINT statement naar de schijf worden geschreven, zelfs op een andere diskette. Zonder CLOSE gebeuren er onvoorspelbare dingen en al heel wat pseudo-professionele programma's kwamen in moeilijkheden door een verkeerd gebruik van dit statement.

```
30 PRINT D$;"CLOSE TRUC"
```

Je mag ook uitsluitend PRINT D\$;"CLOSE" schrijven, maar onthoudt, dat alle geopende files dan afgesloten worden. Je kunt namelijk meer files tegelijk open hebben (zie verder: MAXFILES).

RUN dit programma en geef een CATALOG; tenslotte LIST, zodat we straks weer verder kunnen.

Nu willen wij de file teruglezen. Voer daarom in:

```
40 PRINT D$;"OPEN TRUC"  
45 PRINT D$;"READ TRUC"  
50 INPUT A$  
60 PRINT D$;"CLOSE"  
/O PRINT A$:END
```

Analyseer dit programma en SAVE het even op de diskette, b.v. SAVE SEQTEST. Laat het programma ook werken. Wanneer er in een programma gevraagd wordt naar meer INPUT uit de file, dan er aan gegevens aanwezig is, dan treedt een END OF DATA foutmelding op.

Het is belangrijk ervoor te zorgen, dat fouten, die ontstaan tijdens het wegschrijven van gegevens en het uitlezen ervan, worden opgevangen door een CLOSE commando. Bij het wegschrijven van gegevens naar de diskette kan je een eindteken ingeven, b.v. het woord EINDE. Bij het lezen van de file kan je het INPUT-commando laten volgen door een IF - THEN statement, dat de aanwezigheid van het woord EINDE test en daarna de file CLOSED.

Ook kan het statement ONERR - GOTO voor het INPUT commando worden geplaatst, dat verwijst naar de CLOSE. Vergeet daarna niet de ONERR-flag te resetten door een POKE 216,0.

Oefening: voer in de file TRUC 5 data-regels in en lees die vervolgens weer uit. Verander daarna uitsluitend de dataregels en lees ze weer uit. Wat merk je op!

Oefening: voer regel 17 PRINT D\$;"DELETE TRUC"
en 18 PRINT D\$;"OPEN TRUC" in.
vergelijk de werkwijze van beide oefeningen.

Bij het verwerken van getallen, maar ook zinnen (strings) met een comma erin, moet je goed uitkijken. Wil je b.v. een rij getallen naar de diskette schrijven en doe je dat als volgt:

```
25 PRINT 10,12,14,225,999
```

dan zal de APPLE II bij het teruglezen op een moeilijkheid stuiten. De comma staat nml. in relatie met de TAB-fields (zie aldaar). Om aan de eigenaardige getalsverdeling een eind te maken, zul je elk getal met een afzonderlijk PRINT statement weg moeten schrijven naar de DISK II. De oorzaak is, dat APPLESOFT slechts het Return-karakter (CHR\$(13)) als separator aanmerkt.

Het DOS springt ook vreemd om met GET. Wie een file wil teruglezen, of wegschrijven met GET zal ontdekken, dat het eerst ingevoerde karakter wordt overgeslagen. Het is daarom zaak het eerste karakter willekeurig te kiezen en daarna pas "normaal" met GET verder te werken. Overigens dient GET altijd door PRINT gevolgd te worden, omdat anders onvoorspelbare dingen gebeuren.

Toevoegen van gegevens aan een sequentieel bestand gebeurt met het APPEND commando. APPEND komt dan in de plaats van OPEN. De file pointer wordt daarbij opgeschoven naar het eerste, niet in gebruik zijnde teken aan het einde van de file. APPEND zet de file pointer dus naar het eind van de file, terwijl OPEN zich op het begin ervan richt. Het APPEND commando wordt gevolgd door de aan te roepen filenaam, optioneel het Drivenummer, Slotnummer en Volumenummer.

Let Op: Bij DOS-versie 3.2 (.1) treedt soms een fout aan het licht, die gruwelijke gevolgen voor het bestand kan hebben. APPEND begint dan onverwacht aan het begin van de file!!! Er is een eenvoudige oplossing voor dit probleem, die weggePOKEd kan worden en die juist voor de CLOSE even geCALLED moet worden. Oh, wat een compjoeterbargoens alweer Ja, we worden steeds professioneler! Hier volgt het programma:

```
1 POKE 768,169
2 POKE 769,0
3 POKE 770,76
4 POKE 771,237
5 POKE 772,253
```

Een CALL 768 in regel 57, vlak voor CLOSE, is voldoende, mits een eventuele fout-afhandelingsroutine ernaar verwijst. Het programma plaatst een "end-of-file-marker" (een 0) aan het einde van de file. Het programma POKEd naar adres 768 de volgende machinetaal- instructie: laadt de accumulator met 0 en jump naar de monitor routine \$FDED voor karakteroutput.

POSITION is een bruikbaar commando, om een bepaald veld in een sequentieel bestand aan te vullen. POSITION <filenaam>,R13 is de juiste structuur, om Relatief veld 13 te overschrijven. De velden worden immers gemarkeerd door een Return-code. De computer telt nu het aantal velden, dat moet worden overgeslagen, te beginnen met veld nummer 0.

B. RANDOM ACCESS BESTANDEN

Veruit het meest interessant zijn de willekeurig toegankelijke data-bestanden. Het "random access" bestand (spreek uit: rendum ekses) is het beste te vergelijken met een kaartenbak. Het sequentiele bestand in verhouding daartoe met een telefoonklapper. De random-access file kent het begrip RECORD's.

Een RECORD kan onderverdeeld zijn in FIELD's. Laten we voor een duidelijker begripsbepaling de R.A.-file eens vergelijken met een kaartenbak:

R.A.FILE

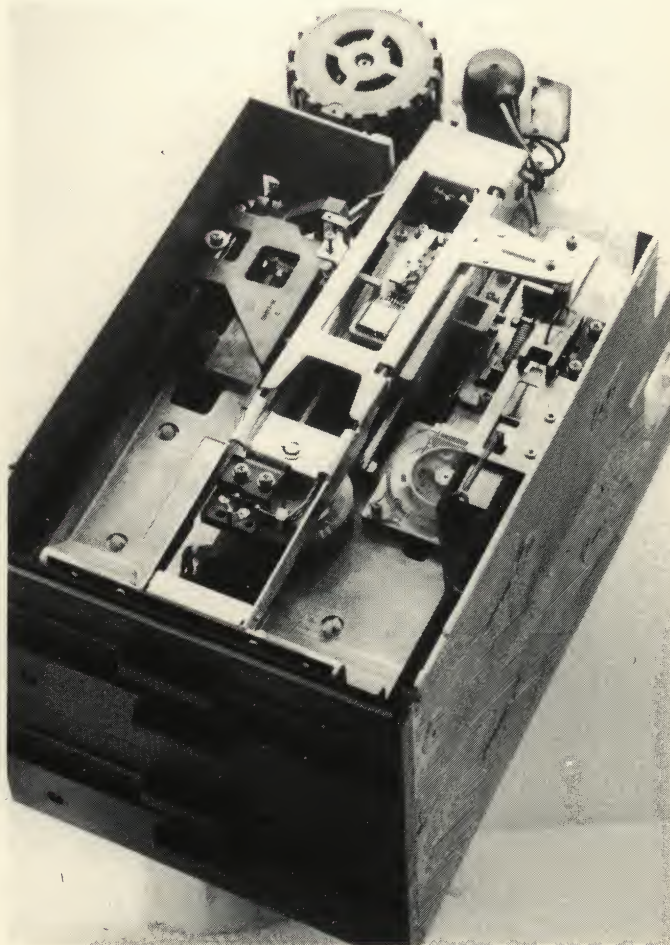
bestandsnaam
record
field
id-code

KAARTENBAK

verzameling kaarten
de enkele kaart
regel op een kaart
kenmerk van de regel, b.v.
uitsluitend te beschrijven
met de woonplaats, het adres

Veelbetekenend is nu de aanwezigheid van de LENGTE-wijzer
bij de definiering van een R.A.-file.

10 PRINT D\$;"OPEN RANAAM,L10,S6,D2,V100"



De lengteparameter is een vereiste, het slotnummer etc. is alleen noodzakelijk wanneer een speciale toepassing is gewenst. Zo kun je je gegevens op een andere schijf wegschrijven en ze gescheiden houden van het feitelijke programma. De lengte L bepaalt de lengte van de record. Het programma moet zo zijn geschreven, dat de fysieke lengte van de records, inclusief Returns etc., de opgegeven waarde niet overschrijdt. Gebeurt dit toch, dan ontstaat een ware puinhoop!

READ en WRITE vereisen de aanduiding voor het Recordnummer in de file; dus: welke kaart er beschreven of gelezen moet worden.

READ RANAAM,R8 of WRITE RANAAM,R8 voert deze truc uit. Het frappante is, dat hieraan tevens Slot- en Drivenummer mogen worden toegevoegd.

Over CLOSE behoeven we niet veel meer te zeggen. Het is van vitale betekenis, dat alle geopende files -zeker bij fouten- ook weer worden afgesloten. Het vergeten van een CLOSE statement kan vergaande gevolgen hebben, die niet tot 1 diskette beperkt blijven.

Een wat mysterieus commando is BYTE. De 'B'-parameter mag worden toegevoegd aan READ, WRITE en POSITION. Zo zal van de 15e record de 5e byte gelezen worden:

```
READ RANAAM,R14,B4
```

Oefening: verklaar waarom ! Denk eraan, waar je met tellen begint.

Oefening: bestudeer de R.A.-demonstratie op de APPLE masterdiskette. Lees het DOS-handboek erop na.

Ten aanzien van het gebruik van 8 inch diskteststations, de grotere broers van de kleine DISK II drives van APPLE, geldt nog een opmerking. De (semi-)professionele gebruiker van de APPLE heeft meestal behoefte aan een grotere gegevensopslag capaciteit en benut daarom de mogelijkheden, die de grotere diskdrives bieden. Het probleem met 8-inch drives is, dat er vaak sprake is van een pseudo-compatibiliteit met het APPLE Mpu systeem. Er ontstaan dan al snel fouten in het wegschrijven of lezen van gegevens, soms moeten hele programma's worden aangepast. Zoals al eerder genoemd is een volstrekt compatibel systeem het 8-inch MultiByte concept, nota bene van Nederlandse bodem. Dit diskdrivesysteem werkt met speciale Amerikaanse interfacekaarten, die het volkomen doen aansluiten bij het APPLE DOS.

Een versie van de MultiBytes is evenwel uitgerust met een hyper- moderne interface, de Sorrento Valley Ass. SVX-4. Hiermee kunnen allerlei diskformaten worden gelezen in z.g. single en double density. Double density wil zeggen, dat er 2 maal zoveel informatie naar de diskette kan worden geschreven door een trucje met de invulling van sectoren. Bij dit systeem is het werken met de VOLUME-parameter van essentiële betekenis. Evenals DRIVE, SLOT, is VOLUME als keuzemaatstaf een in theorie "toegevoegd" stuurcommando.

Het lijkt wat overbodig. Niet echter voor supergrote gegevensopslagmedia, zoals de MultiBytes of de (b.v.) Corvus Winchester Hard Disk stations. Deze vragen onder omstandigheden (afhankelijk van hun interfacing) om een expliciete bepaling van het Volumenummer.

8. EXEC en andere DOS COMMANDO'S

Dat je met je APPLE computer bijna kunt toveren, staat wel vast. Maar met het EXEC commando heeft het er veel van weg, dat je APPLE bovennormaal begaafd is! EXEC staat voor Execute (spreek uit: eksukjoet). Niet in de betekenis van "doodschieten", maar van "voer blindelings uit". En dat is precies wat je APPLE onder EXEC-controle doet. Laten we eens experimenteren:

```
10 REM EXEC TEST
20 PRINT"HET WERKT"
30 END
]SAVE TEST
```

De diskdrive gaat aan en het programma TEST wordt opgenomen.

Tik nu:

```
]FP
10 REM EXEC PROGRAMMA
20 PRINT CHR$(4);"OPEN PROG"
30 PRINT CHR$(4);"WRITE PROG"
32 REM CHR$(4)=CTRL-D
40 PRINT"RUN TEST"
50 PRINT"CATALOG"
60 PRINT"LIST40,60
70 PRINT CHR$(4);"CLOSE PROG"
```

En nu komt het!

]RUN

De diskdrive gaat brommen en PROG wordt vastgelegd - als z.g. textfile (predikaat: T). Tik vervolgens EXEC PROG en kijk wat er allemaal gebeurt.

Omdat de "programma-textfile" PROG inderdaad een sequentiele textfile is, luidt het formaat voor het EXEC commando principieel als volgt:

EXEC (bestandsnaam),R(no.),Slot,Drive,Volume

Probeer:]EXEC PROG,R2

EXEC opent de file en plaatst de startpointer aan het begin van het sequentiele bestand. De R parameter verwijst naar het betreffende relatieve veld binnen de file. R no. 0 is het record, waarnaar de pointer vervalt (z.g. default value), als er niets of niet anders wordt opgegeven. Wanneer het EXEC commando wordt gegeven, dan opent de computer het aangegeven bestand, voert automatisch een READ en een INPUT uit, waarbij de computer reageert ALSOF DE REGELS VIA HET TOETSENBOORD WERDEN INGETIKT!!!

Zo kan een tikfout worden gevonden en afgestraft met de veel-voorkomende kreet: SYNTAX ERR(OR).

EXEC is een prima hulpmiddel voor het snel invoegen van subroutines in programma's. De subroutines worden dan opgenomen als textfiles en terug ingelezen in het te maken programma.

EXEC is een sadistisch commando voor liefhebbers van z.g. programma-beveiligingen... Stel je een programma voor, dat je door een kennis maar eenmaal gedraaid wil hebben! Stel je voor, dat de EXECfile dit programma na een RUN inderdaad DELETE!

EXEC kun je daartoe tot op-bootprogramma maken (een heel speciaal "HELLO"-programma). Probeer eens het volgende, dat bestemd is voor 48K APPLES onder DOS 3.3:

```
]CALL-151
*9E42:14 (return)
*9DBFG (return)
]
```

Doe nu een nieuwe diskette in de diskdrive en tik:

```
10 REM INIT
20 END
```

Let nu goed op: INIT HALLO

Is de diskette met dit vreemde programma geïntialiseerd, dan tikken we het volgende in:

```
]FP
10 PRINT CHR$(4);"OPEN HELLO"
20 PRINT CHR$(4);"WRITE HELLO"
30 PRINT"CATALOG"
40 PRINT"KLAAR"
50 PRINT CHR$(4);"CLOSE HELLO"
]RUN
```

Nu komt de grote wisseltruc.

```
]RENAME HALLO,WEG
DELETE WEG
```

Nu invoeren:

```
]RENAME HELLO,HALLO
```

Start nu de diskdrive opnieuw op met de nieuwe diskette: PR#6.

Als alles goed is, boot het DOS op met de EXECfile HALLO...!

Insgelijks is het mogelijk de APPLE II te laten starten met een BRUN (binary) programma. Echter moet adres *9E42 dan worden gewijzigd in :34.

Natuurlijk zijn er enkele eigenaardigheden met het EXEC commando. Na een onderbreking d.m.v. CTRL-C zal EXEC zijn loop niet hervatten. INPUT statements in een programma worden beschouwd als wijzers naar het volgende veld in de

textfile. Indien het EXEC programma een CLOSE bevat, wordt die genegeerd. Oppassen dus.

Een van de aantrekkelijkste mogelijkheden van EXEC is, dat het ermee mogelijk wordt Integer Basic programma's om te zetten naar APPLESOFT Basic. Via een textfile kan het Integer programma worden omgezet in APPLESOFT, hoewel daarna natuurlijk syntactische problemen moeten worden opgelost. Zie daarvoor de in dit boek aanwezige omzettingstabel.

MAXFILES

Dit commando bepaalt het aantal files, dat tegelijk OPEN kan zijn. Ieder bestand krijgt 595 bytes als geheugenbuffer; waarvan 256 bytes voor READ-operaties en 256 bytes voor WRITES. Het resterend aantal bytes is zuiver voor besturingsdoeleinden. Wanneer het DOS aanwezig is, vervalt het aantal buffers naar 3. Dus MAXFILES heeft als default-value : 3.

Is een DOS-commando in werking, b.v. APPEND, of CATALOG, dan wordt daardoor 1 buffer gebruikt. Je kunt evenwel meer files tegelijk OPEN hebben. De APPLE laat toe, dat MAXFILES tussen 1 en 16 mag liggen. De waarde 16 is arbitrair, omdat meestal niet voldoende geheugen meer aanwezig is. Het zijn geen "kleine" programma's, die 16 buffers nodig hebben

MAXFILES kan vanuit het programma worden geset. Het wordt dan vooraf gegaan door CTRL-D. Er is echter een probleem! MAXFILES verschuift de HIMEM: pointer en vernietigt een Integer Basic programma, indien het vanuit Integer Basic programmatuur wordt ingesteld. Ook "clobbered" het Applesoft strings, indien in de loop van een Applesoft programma MAXFILES wordt uitgebreid. Het is zaak MAXFILES vooraf te setten.

MON

Het MON commando staat voor MONITOR. Het laat toe, dat je op de TV-monitor kunt zien welke READ en WRITE operaties worden uitgevoerd. Maar je moet dan wel de juiste parameters aangeven.

MON C	toont de diskcommando's.
MON I	toont de disk-input
MON O	toont de disk-output
MONC,I,O	is het samenvattende commando en probeer het eens uit.

Het wordt (natuurlijk) voorafgegaan door CTRL-D. Meestal is het de eerste regel van je programma, indien je de disk-operaties wilt volgen.

NOMON

Het tegenovergestelde van MON. Ook dit commando zal het eerste van je programma zijn. MON en NOMON zijn in feite programmeerhulpjes.

TRACE

Dit commando is eveneens een programmeerhulp. Het laat zien welke regelstappen een programma neemt, maar zonder Return's. DOS commando's worden vaak niet overgenomen en veroorzaken zelfs een gestoord verloop van de TRACE functie. De remedie is simpel.

Vervang de benoeming van de stringvariabele D\$, die door veel programmeurs gebruikt wordt als codering voor CTRL-D (CHR\$(4)), door D\$=CHR\$(13)+CHR\$(4). Deze truc is voldoende, om de problemen het hoofd te bieden.

BSAVE filenaam,A111,L111,<S,D,V>

Om grafische beelden en machinetaalprogramma's te kunnen opslaan op de diskette, gebruiken wij het commando BSAVE. Dit commando is tamelijk gecompliceerd. Afgezien van het benoemen van de filenaam moet je ook het startadres en de lengte van het programma aangeven. Dit mag in decimale- en hexadecimale notatie plaats- vinden. Echter niet door elkaar. Indien de hexadecimale notatie wordt gebruikt, dan worden A en L gevolgd door het \$-teken.

De B van BSAVE staat voor binair, hetgeen al enigszins aangeeft, dat we op machinetaalniveau bezig zijn. De adreswaarde A zal niet hoger dan 65535 (dec) of \$FF FF (hex) kunnen zijn. De L parameter mag evenwel niet groter zijn dan 32767 (dec), of \$7F FF (hex). Dat komt door beperkingen van het DOS. Wil je een langer programma opslaan, dan zijn daarvoor 2 BSAVE's nodig.

Om de hexadecimale start- en lengtewaarde te kunnen vinden, is een tweetal geheugenplaatsen aan te roepen, vermits het te BSAVEN programma eerder met BLOAD werd ingeladen. Tik het volgende:

```
CALL-151
*AA72.AA73 AA60.AA61 (precies overnemen, incl.spatie)
*9DBFG
]
```

Op AA72/73 verschijnt de A-waarde en L op AA60/61, alles in hex. Dit geldt voor een 48K APPLE II.

BLOAD filenaam,<A111,S,D,V>

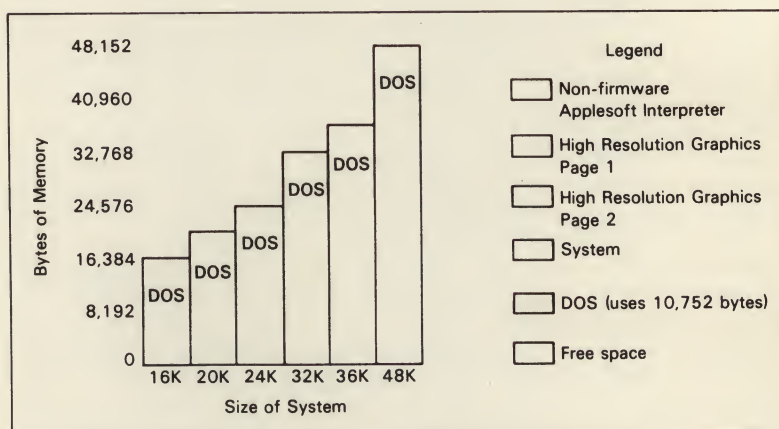
Laadt een binair programma in het geheugen. Dit programma kan vervolgens vanuit Applesoft worden aangeroepen door de CALL en USR functies. Machinetaalprogramma's weigeren dienst, indien ze op een onjuiste plaats in het geheugen worden geladen. Wordt de A-waarde niet vermeld, dan komt de file in het geheugen op de plaats vanwaar het werd geBSAVED. In tegenstelling tot het LOAD statement wordt bij BLOAD uitsluitend het geheugengedeelte, waarnaar de machinetaalroutines worden verwezen, beïnvloed.

BRUN filenaam,<A111,S,D,V>

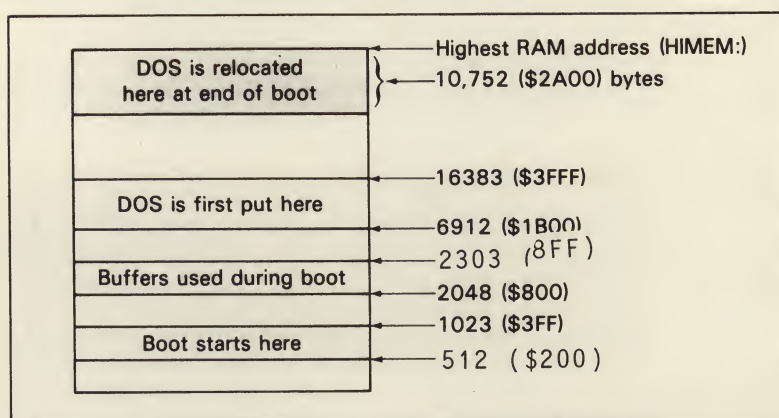
Voert automatisch een JMP uit naar het startadres van het machinetaalprogramma. BRUN mag nooit worden gebruikt bij grafische afbeeldingen, omdat er dan onverwachte dingen kunnen gebeuren.

CHAIN filenaam, D,S,V

Uitsluitend bestemd voor Integer Basic. RUNned een Integer programma zonder eerder gebruikte waarden van arrays of variabelen te veranderen. Zie hieromtrent het DOS-Manual.



Read/Write Memory (RAM) Usage



Memory Used During DOS Boot

8. D O S Lees/Schrijf Operaties

Eerder in dit hoofdstuk hebben we al even kennis gemaakt met diskettes. Andere namen voor diskettes zijn flexibele schijven of "floppy disks", deze laatste meestal afgekort tot "floppie". Ook werd melding gemaakt van het feit dat er twee maten hiervan zijn, n.l. de 5 1/4 inch en de 8 inch diskettes.

In de hier gepresenteerde beschrijving van de opbouw van een diskette en de daarop aanwezige systeem-informatie gaan wij uit van het 5 1/4 inch formaat.

Diskettes worden op een bepaalde manier geformatteerd. Deze formattering verschilt van computer-systeem tot computer-systeem en van computer-type tot computer-type. Ons beperkend tot het APPLE - systeem komen we al twee formatteringen tegen, in de wandeling DOS 3.2 en DOS 3.3 genoemd.

Hoewel er meer verschillen zijn tussen de twee genoemde DOSsen volstaan we hier met het noemen van afwijkende diskette-formattering en de gevolgen hiervan op de gegevensopslag.

De tabel geeft de voornaamste verschillen :

Schijf - Opbouw			

	3.2	*	3.3

		*	
Aantal Sporen	35	*	35
		*	
Sectoren per Spoor	13	*	16
		*	
Sectoren per Diskette	455	*	560
		*	
Bytes per sector	256	*	256
		*	
Bytes per Diskette	116480	*	143360
		*	

		*	
Sectoren voor data	403	*	496
		*	
Bytes voor data	103168	*	126976
		*	

De onderverdeling van een diskette in Sporen (Tracks) en Sectoren (Sectors) veronderstellen wij als bekend. Mocht het je even ontschoten zijn, is het raadzaam de tekst op blz. 9-2 en de tekening op blz. 9-3 te raadplegen.

We weten nu dat met DOS 3.3 $35 \times 16 = 560$ sectoren beschikbaar zijn. Voor de Computergebruiker zijn echter maar 496 sectoren beschikbaar.

De andere 64 sectoren worden door het Computer-Systeem gebruikt voor opslag van Software en Gegevens voor de besturing van de Disk-Drive. Van deze 64 sectoren worden er 48 gebruikt voor het opslaan van het eigenlijke Disk Operating System (DOS) en de rest als een Catalogus met daarin alle informatie over de op schijf aanwezige Programmatuur en Gegevensbestanden. De Sporen die voor de opslag van deze informatie gebruikt worden zijn Tracks 0,1,2 en 17.

Het Disk Operating Systeem moet dus rekening houden met Sporen waar Systeem-programmatuur is opgeslagen (DOS), Sporen waar Gebruikersprogramma's staan, weer andere Sporen met Systeem-bestanden ("Disk-Directory") plus nog Sporen met Gebruikersbestanden ("Data-Files"). Alsof dit nog niet genoeg is, wordt ook het bepalen van het type van de "File" (Integer, Applesoft, enz.) nog aan het DOS overgelaten. Dit alles bij elkaar vereist een goede administratie. En, aangezien we probleemloos van de Disk-Drive gebruik maken is dat probleem toch afdoende opgelost.

Diskette toegang door het D.O.S. :

In deze sectie zullen we het hele proces dat het DOS ondergaat alvorens een Data-sector van diskette in te lezen, van stap tot stap gaan volgen. Een uiterst handig hulpmiddel voor het bestuderen van dit proces is het Super Sector Editor Programma van de heer Paul Spee. Dit programma, (een regelrechte "must" voor de Disk-Drive bezittende Apple Programmeur) is verkrijgbaar bij de Hobby Computer Club en wel de afdeling Software-service van de Apple Gebruikers Groep Nederland. Je vindt het op Schijf 26 en kost rond F 20,=.

Omdat begripelijkheid bovenaan staat in dit Handboek hebben wij als voorbeeld de Master-Diskette genomen die bij aankoop van een Disk-Drive wordt bijgeleverd. De keus viel hierop, omdat deze diskette bij zeer vele Apple-bezitters in de Software Bibliotheek is te vinden.

Algemeen :
=====

Het Disk Operating System werkt volgens een aantal conventies. Om er een paar te noemen :

1. DOS verwacht VTOC en CATALOG op Track 17 (Hex 11)
2. DOS schrijft altijd eerst naar de Tracks boven Track 17. T.w. 18 t/m 34.
3. Zijn deze "vol", dan worden Tracks 16 t/m 4 beschreven.
4. Uitbreiding van het aantal sectoren van een bestand of

programma zal eerst aan de Track van dat bestand of programma worden toegekend. Is hiervoor geen plaats, dan zal een nieuwe Track worden gebruikt.

Het DOS zal dus ook nog een administratie bij moeten houden met al dan niet bezette sectoren van de individuele Tracks. Welnu, deze administratie wordt bijgehouden in de Volume Table of Contents (VTOC).

De VTOC :

=====

Deze vinden we op Track 17, Sector 0. In hexadecimale notatie \$11,0.

Voor de naam Volume Table of Contents is geen goede Nederlandse naam te vinden.

Misschien dat de benaming Inhoud en Omvang Tabel het meest in de buurt komt.

De VTOC (we zullen deze naam maar blijven gebruiken) bevat een overzicht van alle bezette - en niet bezette sectoren van de afzonderlijke sporen.

Aangezien DOS een schijf met complete sectoren van 256 bytes beschrijft, kent de VTOC geen half-bezette sectoren. Wordt een sector maar voor 1/3 of 1/4 volgeschreven, dan zal die gehele sector als "vol" worden beschouwd en als zodanig in de VTOC worden opgenomen. Voor elk Spoor zijn 4 bytes gereserveerd en al deze bytes worden achter elkaar beschreven. Elke groep van 4 bytes wordt een Track Bit Map genoemd. Twee van deze 4 bytes worden echter maar gebruikt, de resterende 2 zijn reserve.

Nu zijn 2 bytes nog altijd 16 bits en dat is weer het aantal sectoren van een Spoor. Welnu, voor elke "volle" sector wordt een binaire 0 in het corresponderende bit gezet. Is de betreffende sector "leeg", dan zal een binaire 1 in dat bit staan.

Een voorbeeld van een Track Bit Map vinden we op blz. 134 van het DOS Manual.

Behalve de 35 Track Bit Maps vinden we ook nog algemene informatie omtrent de Diskette in de VTOC.

De opbouw van de VTOC ziet er zo uit:

Byte 0 (\$00)	Wordt niet gebruikt
1 (\$01)	Geeft Spoor-nummer van eerste "Directory" sector
2 (\$02)	Geeft Sector-nummer van eerste "Directory" sector
3 (\$03)	DOS versie-nummer
4 (\$04)	Wordt niet gebruikt
5 (\$05)	Wordt niet gebruikt
6 (\$06)	Diskette Volume Nummer
7 (\$07)	Wordt niet gebruikt
/	
/	
/	
38 (\$26)	Wordt niet gebruikt
39 (\$27)	Aantal Tracks/Sectors in de Track/Sector List

40 (\$28)	Wordt niet gebruikt
/	
/	
/	
47 (\$2F)	Wordt niet gebruikt
48 (\$30)	Eerstvolgend te gebruiken Spoor
49 (\$31)	Richting van Spoortoekenning
50-51 (\$32-\$33)	Wordt niet gebruikt
52 (\$34)	Aantal Sporen per Diskette
53 (\$35)	Aantal Sectoren per Diskette
54 (\$36)	Aantal bytes per sector , laag
55 (\$37)	Aantal bytes per sector , hoog
56-59 (\$38-\$3B)	Spoor 0 Bit Map
60-63 (\$3C-\$3F)	Spoor 1 Bit Map
64-67 (\$40-\$43)	Spoor 2 Bit Map
68-71 (\$44-\$47)	Spoor 3 Bit Map
/	/
/	/
/	/
/	/
192-195 (\$C0-\$C3)	Spoor 22 Bit Map
196-255 (\$C4-\$FF)	Wordt niet gebruikt

Voor de hierna volgende beschrijving gaan we ervan uit dat je over het Super Sector Editor programma (of een soortgelijk programma) beschikt.

Laadt SSE in je systeem en doe de Apple Master Diskette in je Disk-Drive. (Het is verstandig om voor alle zekerheid hiervoor een kopie te gebruiken. Deze kan je maken door gebruik te maken van het COPY programma op dezelfde Master-Schijf).

Met de SSE-instructie 11,0R wordt de VTOC van schijf ingelezen. Met een P-instructie wordt deze informatie afgedrukt.

Onderbreek het programma na een 15tal regels door op de spatiebalk te drukken.

De informatie die nu op je beeldscherm staat moet ongeveer overeenkomen met Figuur 1.

Laten we de lijst eens doorlopen.

Byte \$00 slaan we over. Op \$01 en \$02 vinden we de waarde 11 en 0F. Dit is dus de eerste "Directory" Track/Sector. Snel omrekenen leert ons dat dit decimaal Track 17, Sector 15 moet zijn. En in 9,9 van de 10 gevallen vinden we hier inderdaad de eerste Catalog-sector van een diskette.

Voor de nieuwsgierigen : sommige doorgewinterde programmeurs zetten de Catalog op een andere Track om copieren van hun diskette te verhinderen of bemoeilijken. Een woord van waarschuwing is hier tevens op zijn plaats:

```
*****
* PROBEER NIET ZONDER MEER ZELF MET DE *
* CATALOG TE KNOEIEN.                  *
* DIT KAN CATASTROFALE GEVOLGEN HEBBEN *
*****
```


Byte \$03 geeft ons het DOS versie nummer , waarden die we hier vinden kunnen 03 of 02 zijn.

Byte \$06 is het Diskette-Volume nummer. Dit kan variëren tussen 01 en FE (1-254 Decimaal).

Het Volume nummer kan gebruikt worden voor bescherming van Diskettes. Je kan je voorstellen dat een bepaald bestand (dit kan een adressenlijst of een Voorraadoverzicht zijn) meerdere diskettes in beslag neemt. Door aan deze diskettes een uniek Volumenummer te geven worden de gegevens al op programmaniveau beschermd. Als het programma dit Volumenummer opgeeft zal nooit een verkeerd bestand gemuteerd kunnen worden.

Een ander, treffend , voorbeeld is dat van een dagelijks transactie-file. Hierin worden gegevens geschreven die betrekking hebben op een bepaalde dag, b.v. Maandag. Stel je nu even voor dat je voor elke dag van de week een andere Diskette wilt gebruiken (b.v. om een duidelijk dagoverzicht te verkrijgen). Door voor elke dag een ander Volumenummer toe te kenne zullen de gegevens van Dinsdag nooit op een Diskette van Maandag worden geschreven.

Probeer je dat toch dan zal de foutmelding VOLUME MISMATCH worden gegeven.

Het gebruik van een Volumenummer in deze vorm gaat alleen op voor de "normale" 5 1/4 inch diskettes en sommige 8 inch Diskette Sub Systemen. Meer informatie hierover is te vinden op blz. 9-9.

Voor andere 8 inch Systemen en de grote Hard Disk systemen wordt het Volumenummer gebruikt om een een bepaald gebied op die Disks aan te duiden. Bij een 5 Megabyte Hard Disk kunnen dan b.v. 4 Volumes worden geadresseerd. Ben je in het bezit van zo'n systeem, lees dan de bijgeleverde Documentatie er eens op na.

Nu komt er een tijd niets en vinden we de volgende informatie op byte \$27. Dit is een vaste waarde en geeft het maximum aantal Spoor/Sector paren per Track/Sector List (zie aldaar). Deze waarde is 7A of 122.

Weer loze ruimte en komen we bij byte \$30. In dit byte wordt de eerst volgende te gebruiken Track aangegeven.

Byte \$31 geeft de richting van Track toekenning aan.

Wanneer hier de waarde 0F staat wordt in opgaande richting Tracks toegekend, d.w.z. van Track 18 oplopend naar Track 35. Vinden we hier de waarde FF dan wordt in neergaande richting de Tracks toegekend, dus vanaf Track 16 tot Track 3. Dit is in tegenstelling van wat er in het DOS MANUAL staat. Daar wordt gesproken over een Track Bit Maskerings Map.

Bytes \$34 en \$35 geven het aantal sporen en sectoren per diskette. Meestal is dit 23(hex) of 35, resp. 10(hex) of 16.

PAS OP !

Het DOS MANUAL geeft op blz. 132 voor het aantal sectoren de waarde 0F(hex), dit zouden er 15 zijn. De VTOC geeft het correct aantal aan, n.l. 10(hex) ofwel 16. De nummering (sec!) loopt echter van 00(hex) tot 0F(hex).

Bytes \$36 en \$37 geven het aantal bytes per sector die laag resp. hoog zijn.
Vanaf byte \$38 krijgen we dan de 35 Track Bit Maps, die zoals nu bekend, ieder 4 bytes lang zijn.
De resterende bytes \$C4 tot \$FF worden niet gebruikt.
Indien er meer dan 35 Sporen op de Diskette voor zouden komen, kunnen de Track Bit Maps van deze extra sporen hier opgeborgten worden.

We hebben gezien dat het DOS veel algemene informatie in de VTOC kan vinden. Meer specifieke informatie wordt gevonden in de DIRECTORY. Dit is dan de volgende stap die we maken.

De DIRECTORY :
=====

De DIRECTORY wordt ook wel de CATALOG genoemd en deze tweede benaming geeft al aan wat voor informatie er is te vinden.

Hier wordt een overzicht gegeven van alle op de diskette aanwezige "FILES" (= bestand). Tevens worden hier de FILE-typen en hun status t.a.v. al dan niet uitwisbaarheid bijgehouden.

Volgens de VTOC moeten we de eerste "Directory"-sector op Spoor 17, Sector 15 vinden. Dat is 11 en 0F hexadecimaal, dus moeten de SSE-instructies 11,FR en P Figuur 2 opleveren.

Ook hier weer een gestructureerde opbouw:

Byte 0 (\$00)	Wordt niet gebruikt
1 (\$01)	Track nummer van volgende "Directory"-sector
2 (\$02)	Sector Nummer van volgende "Directory"-sector
3-10 (\$03-\$0A)	Wordt niet gebruikt
11-45 (\$0B-\$2D)	Beschrijving File 1
46-80 (\$2E-\$50)	Beschrijving File 2
81-115 (\$51-\$73)	Beschrijving File 3
116-150 (\$74-\$96)	Beschrijving File 4
151-185 (\$97-\$B9)	Beschrijving File 5
186-220 (\$BA-\$DC)	Beschrijving File 6
221-255 (\$DD-\$FF)	Beschrijving File 7

Bestudering van bovenstaande gegevens levert ons de volgende conclusies op:

1. Er gaan 7 File beschrijvingen in een Directory Sector.

Dat zijn dan $15 \times 7 = 105$ Files in de Directory.
(Pro Memorie ; Sector 0 bevat de VTOC)

2. Door de verwijzing in de bytes \$01 en \$02 weten we dat er "terug geteld" wordt.

3. Elke beschrijving neemt 35 bytes in beslag.

Iedere File beschrijving is op dezelfde manier geformatteerd. Dit formaat ziet er als volgt uit (de byte-nummers zijn RELATIEF, d.w.z. byte 0 is het eerste byte van de specifieke File beschrijving.):

Byte 0	Track nummer van de Track/Sector List van die File
1	Sector nummer van de Track/Sector List van die File
2	File - type
3-20	File - naam
21-22	Aantal gebruikte sectoren

Met een blik op het scherm zijn deze bytes wel te vinden.
Alleen het relatieve byte 2 behoeft nog nadere uitleg.
In dit byte wordt n.l. het File - type en
"status" opgeborgen.

Mogelijke hexadecimale waarden kunnen zijn :

Textfile	0	(niet "gelockt")	of	80	("gelockt")
Integer	1	,,	of	81	,,
Applesoft	2	,,	of	82	,,
Binair	4	,,	of	84	,,

Opmerking :

Diegenen, die in het bezit zijn van de DOS TOOLKIT met de
Relocatable Files kunnen nog twee andere waarden
aantreffen , n.l. 10 (indien niet "gelockt") en 90
(indien wel "gelockt").

Nu we ook de weg weten in de DIRECTORY / CATALOG van' de
diskette , kunnen we de volgende stap maken. En deze
voert ons naar de Track/Sector List.

De Track/Sector List:
=====

De Track/Sector list geeft een overzicht van alle
gebruikte Tracks en hun Sectors voor een individuele
File. Elke File, die in de DIRECTORY voorkomt, krijgt een
complete sector toegewezen voor deze informatie. Indien
het bestand zo groot is, dat er zoveel sectoren worden
gebruikt dat deze informatie niet in een Track/Sector
List Sector kan worden opgeslagen, wordt er een tweede
Sector daarvoor aangewezen. Uiteraard zal dan een
verwijzing naar deze sector worden opgenomen in de eerste
sector.

Een typische sector van een Track/Sector List heeft de
volgende structuur:

Byte 0 (\$00)	Wordt niet gebruikt
1 (\$01)	Track nummer van volgende TSL-Sector
2 (\$02)	Sector nummer van volgende TSL-Sector

=> Indien bytes \$01 en \$02 de waarde 00 bevat, dan
is slechts 1 sector nodig <=

3 (\$03)	Wordt niet gebruikt
4 (\$04)	Wordt niet gebruikt
5-6 (\$05-\$06)	Sector Basis nummer (telt groepen van 122 sectoren)
7-11 (\$07-\$0B)	Wordt niet gebruikt
12 (\$0C)	Track nummer van eerste File sector
13 (\$0D)	Sector nummer van eerste File sector
14 (\$0E)	Track nummer van tweede File sector
15 (\$0F)	Sector nummer van tweede File Sector
// //	//
// //	//
// //	//
254 (\$FE)	Track nummer van 122ste File sector
255 (\$FF)	Sector nummer van 122ste File sector

Met behulp van de gegevens in de DIRECTORY zijn we nu in staat om de TSL van de File HELLO op het beeldscherm te zetten. Deze moet overeenkomen met Figuur 3. Probeer het maar eens.

Er niet uitgekomen? Geen probleem, hoor, alle begin is tenslotte moeilijk.

Laten we m.b.v. de SSE nog eens de eerste DIRECTORY sector inlezen. (SSE-instructies 11, FR en P). Nu zoeken we de eerste "entry" op, deze moet beginnen bij byte \$0B en doorlopen tot byte \$2E. Verder weten we dat de relatieve bytes 0 en 1 de Track en Sector van de TSL aangeven. Dus moeten de bytes \$0B en \$0C dit aangeven. De waarden die we hier vinden zijn 13 en 0F. Door deze Sector in te lezen hebben we de TSL van het HELLO programma.

Probeer nu ook eens de TSL van het programma ANIMALS te vinden. Doe hetzelfde voor het programma MUFFIN. Lukt dat de eerste keer nog niet, lees dan het gedeelte over de DIRECTORY/CATALOG nog eens na.

DATA SECTOREN:

=====

Hoe een Data-sector gevonden kan worden behoeft hier geen verdere uitleg. Daarom nog wat opmerkingen van algemene aard.

De opbouw van een Data-sector verschilt van File-type tot File-type. Dit houdt verband met de eigenaardigheden van de APPLESOFT - en INTEGER BASIC taal en de structuren van een Text-file en Binary-file.

1. TEXT-FILE Sector:

Deze sectoren worden sequentieel beschreven. De individuele records worden van elkaar gescheiden door een CARRIAGE RETURN (8D hexadecimaal). 00 Wordt beschouwd als een End Of File Marker. (zie figuur 4)

2. BINARY-FILE SECTOR:

Een binaire file is een kopie van een aantal geheugenplaatsen op Diskette. Deze worden voorafgegaan door het startadres en de lengte. We kennen deze parameters al omdat zij bij een BSAVE commando als A\$aaa,L\$lll worden meegegeven, waar aaa het startadres en lll de lengte is. (zie figuur 5)

3. APPLESOFT PROGRAM SECTOR:

De eerste beschreven sector begint met de programmalengte in bytes, gevolgd door het programma. (zie figuur 6)

4. INTEGER BASIC PROGRAM SECTOR:

De eerste beschreven sector begint met de programmalengte, gevolgd door de lijnlengte en lijnnummers, daarna het programma en afgesloten met \$51. De hexadecimale waarde 01 geeft aan dat het einde van een lijn is bereikt. (zie figuur 7)

SLOTWOORD:

We hebben het DOS nu gevolgd bij het benaderen van een Diskette. We weten nu waar welke informatie staat. Ook is nu bekend hoe de informatie wordt opgeslagen. Als voorbeeld gebruikten we daarvoor de Master-Diskette van APPLE. Zoals je weet is deze schijf "write-protected", wat inhoudt dat je zelf geen informatie naar die Diskette kan wegschrijven. Dit is er de oorzaak van dat de gegevens zo mooi gestructureerd in de VTOC en DIRECTORY zijn opgeslagen. Bij een Diskette die normaal gebruikt wordt en waarop regelmatig Files worden GESAVED en GEDELET, zal deze structuur zich niet zo snel laten herkennen. Als DOS een file moet wegschrijven, wordt dat gedaan op de eerste de beste vrije plaats die daarvoor in aanmerking komt. M.a.w. het is niet zo dat de eerste DIRECTORY/CATALOG sector perse 7 "entries" bevat.

Waarschijnlijk zal bij sommigen de interesse nu gewekt zijn voor de mogelijkheden van het DOS. Er bestaat literatuur hierover maar is overwegend Engels en soms van een pittig gehalte. Diegenen, die zich hierdoor niet afgeschrikt voelen, kunnen contact opnemen met de uitgever.

Wij besluiten met een herhaling van de eerder gegeven
waarschuwing:

* PROBEER NIET ZONDER MEER ZELF MET DE *
* CATALOG TE KNOEIEN. *
* DIT KAN CATASTROFALE GEVOLGEN HEBBEN *

TRACK/SECTOR LIST

00:	00 00 00 00 00 00 00 00	00000000
08:	00 00 00 00 13 0E 13 0D	0000SNSM
10:	13 0C 13 0B 13 0A 00 00	SLSKSJ00
18:	00 00 00 00 00 00 00 00	00000000
20:	00 00 00 00 00 00 00 00	00000000
28:	00 00 00 00 00 00 00 00	00000000
30:	00 00 00 00 00 00 00 00	00000000
38:	00 00 00 00 00 00 00 00	00000000
40:	00 00 00 00 00 00 00 00	00000000
48:	00 00 00 00 00 00 00 00	00000000
50:	00 00 00 00 00 00 00 00	00000000
58:	00 00 00 00 00 00 00 00	00000000
60:	00 00 00 00 00 00 00 00	00000000
68:	00 00 00 00 00 00 00 00	00000000
70:	00 00 00 00 00 00 00 00	00000000
78:	00 00 00 00 00 00 00 00	00000000
80:	00 00 00 00 00 00 00 00	00000000
88:	00 00 00 00 00 00 00 00	00000000
90:	00 00 00 00 00 00 00 00	00000000
98:	00 00 00 00 00 00 00 00	00000000
A0:	00 00 00 00 00 00 00 00	00000000
AB:	00 00 00 00 00 00 00 00	00000000
B0:	00 00 00 00 00 00 00 00	00000000
BB:	00 00 00 00 00 00 00 00	00000000
C0:	00 00 00 00 00 00 00 00	00000000
CB:	00 00 00 00 00 00 00 00	00000000
D0:	00 00 00 00 00 00 00 00	00000000
DB:	00 00 00 00 00 00 00 00	00000000
E0:	00 00 00 00 00 00 00 00	00000000
EB:	00 00 00 00 00 00 00 00	00000000
F0:	00 00 00 00 00 00 00 00	00000000
FB:	00 00 00 00 00 00 00 00	00000000

Figuur 3

TEKST FILE

00:	CC CF C1 C4 C3 CF CD D0	LOADCOMP
08:	C1 D2 C5 D2 AC C4 B2 BD	ARER,D2M
10:	A0 B0 A0 BF C3 C8 D2 A4	0 ?CHR\$
18:	AB B4 A9 A2 CF D0 C5 CE	(4)"OPEN
20:	C3 CF CD D0 C1 D2 C5 D2	COMPARER
28:	AE D4 AC C4 B2 A2 BA BF	.T,D2":?
30:	C3 C8 D2 A4 AB B4 A9 A2	CHR\$(4)"
38:	D7 D2 C9 D4 C5 C3 CF CD	WRITECOM
40:	D0 C1 D2 C5 D2 AE D4 A2	PARER.T"
48:	BA C8 CF CD C5 BA D0 CF	:HOME:PO
50:	CB C5 A0 B3 B3 AC B3 B3	KE 33,33
58:	BA CC C9 D3 D4 B1 AD BA	:LIST1-:
60:	BF C3 C8 D2 A4 AB B4 A9	?CHR\$(4)
68:	A2 C3 CC CF D3 C5 A2 BA	"CLOSE":
70:	D4 C5 D8 D4 BA C4 C5 CC	TEXT:DEL
78:	B0 AC B0 BA C5 CE C4 BD	0,0:ENDM
80:	D2 D5 CE BD C5 D8 C5 C3	RUNMEXEC
88:	CA D5 CD D0 AC C4 B1 BD	JUMP,D1M
90:	C5 CE C4 BD BD 00 00 00	ENDMM
98:	00 00 00 00 00 00 00 00	
A0:	00 00 00 00 00 00 00 00	
AB:	00 00 00 00 00 00 00 00	
B0:	00 00 00 00 00 00 00 00	
B8:	00 00 00 00 00 00 00 00	
C0:	00 00 00 00 00 00 00 00	
C8:	00 00 00 00 00 00 00 00	
D0:	00 00 00 00 00 00 00 00	
D8:	00 00 00 00 00 00 00 00	
E0:	00 00 00 00 00 00 00 00	
E8:	00 00 00 00 00 00 00 00	
F0:	00 00 00 00 00 00 00 00	
FB:	00 00 00 00 00 00 00 00	

Figuur 4

BINARY (BOOT 13)

00:	00 17 F0 08 20 E3 03 84	WpH cCD
08:	00 85 01 A0 01 B1 00 8D	EA A1M
10:	90 17 C8 B1 00 8D 91 17	PWH1MQW
18:	20 58 FC A0 FF C8 B9 96	X! H9V
20:	17 08 09 80 20 ED FD 28	WHI m}(
28:	10 F3 A9 BF 85 33 20 6A	Ps)?E3 j
30:	FD AD 00 02 C9 8D F0 0F	}-BIMpD
38:	C9 B1 90 DC C9 B8 B0 D8	I1P\I8OX
40:	0A 0A 0A 0A 8D 82 17 A9	JJJJMBW)
48:	17 A0 81 20 00 1D B0 F7	W A 2J0w
50:	AD FE 16 8D 8A 17 85 13	-~VMJWES
58:	E6 13 AD FF 16 4A 4A 4A	fS-VJJJ
60:	85 10 A9 17 A0 B1 20 00	EP)W A 2
68:	1D B0 F7 EE 8A 17 EE 86	J0wnJWnF
70:	17 AD 86 17 C5 10 F0 EA	W-FWEppj
78:	90 E8 AD 82 17 AA A9 00	Ph-BW*)2
80:	85 12 6C 12 00 01 60 01	ER1R2A'A
88:	00 00 00 92 17 00 16 00	222RW2V2
90:	00 01 00 00 C6 FD 00 01	2A22F}2A
98:	EF D8 0D 0D 20 20 20 20	oXMM
A0:	20 20 20 20 20 20 20 31	1
A8:	33 2D 53 45 43 54 4F 52	3-SECTOR
B0:	20 42 4F 4F 54 20 55 54	BOOT UT
B8:	49 4C 49 54 59 0D 0D 53	ILITYMMS
C0:	4C 4F 54 20 54 4F 20 42	LOT TO B
C8:	4F 4F 54 20 46 52 4F 4D	OOT FROM
D0:	20 28 44 45 46 41 55 4C	(DEFAULT
D8:	54 3D 36 29 A0 0D 53 4C	T=6) MSL
E0:	4F 54 20 54 4F 20 42 4F	OT TO BO
E8:	4F 54 20 46 52 4F 4D 20	OT FROM
F0:	28 44 45 46 41 55 4C 54	(DEFAULT
F8:	3D 36 29 A0 A0 A0 A0 A0	=6)

Figuur 5

APPLESOFT

00: 71 04 19 08 0A 00 B2 20	qDYHJ@2
08: 20 2D 2D 20 44 4F 53 20	-- DOS
10: 33 2E 33 20 48 45 4C 4C	3.3 HELL
18: 4F 00 20 08 14 00 B2 20	0@ HT@2
20: 00 2B 08 1E 00 B9 3A BA	@(H^@I::
28: 00 2E 08 2B 00 97 00 59	@.H(@W@Y
30: 08 32 00 BA 22 44 4F 53	H2@:"DOS
38: 20 56 45 52 53 49 4F 4E	VERSION
40: 20 33 2E 33 20 20 20 20	3.3
48: 20 20 20 20 20 20 20 20	
50: 30 38 2F 32 35 2F 38 30	0B/25/B0
58: 22 00 BB 08 3C 00 BA 3A	"@KH<@::
60: BA 22 41 50 50 4C 45 20	: "APPLE
68: 49 49 20 50 4C 55 53 20	II PLUS
70: 4F 52 20 52 4F 4D 43 41	OR ROMCA
78: 52 44 20 20 20 53 59 53	RD SYS
80: 54 45 4D 20 4D 41 53 54	TEM MAST
88: 45 52 22 00 92 08 46 00	ER"@RHF@
90: B2 20 00 B4 08 50 00 B2	2 @4HP@2
98: 20 2D 2D 50 4F 4B 45 20	--POKE
A0: 4C 41 4E 47 55 41 47 45	LANGUAGE
A8: 20 43 41 52 44 20 46 49	CARD FI
B0: 4E 44 45 52 00 05 09 5A	NDER@EIZ
B8: 00 B9 37 36 38 2C 30 3A	@9768,0:
C0: B9 37 36 39 2C 31 37 33	9769,173
C8: 3A B9 37 37 30 2C 30 3A	:9770,0:
D0: B9 37 37 31 2C 32 32 34	9771,224
D8: 3A B9 37 37 32 2C 37 32	:9772,72
E0: 3A B9 37 37 33 2C 31 37	:9773,17
EB: 33 3A B9 37 37 34 2C 31	3:9774,1
F0: 32 39 3A B9 37 37 35 2C	29:9775,
F8: 31 39 32 3A B9 37 37 36	192:9776

Figuur 6

INTEGER

00: 6D 10 08 00 00 5F B1 E8	mPH@_1h
08: 03 01 1E 01 00 5D AA AA	CA^A@]**
10: AA AA AA AA AA AA AA AA	*****
18: AA AA AA AA AA AA AA AA	*****
20: AA AA AA AA AA AA AA 01	*****A
28: 1E 02 00 5D AA A0 A0 A0	^B@]**
30: A0 A0 A0 A0 A0 A0 A0 A0	
38: A0 A0 A0 A0 A0 A0 A0 A0	
40: A0 A0 A0 A0 AA 01 1E 03	*A^C
48: 00 5D AA A0 A0 A0 A0 C1	@]** A
50: CE C9 CD C1 CC D3 BA A0	NIMALS:
58: A0 A0 A0 A0 A0 A0 A0 A0	
60: A0 A0 AA 01 1E 04 00 5D	*A^D@]**
68: AA A0 A0 A0 C3 CF D0 D9	* COPY
70: D2 C9 C7 C8 D4 A0 B1 B9	RIGHT 19
78: B7 BB A0 C2 D9 A0 A0 A0	78 BY
80: AA 01 1E 05 00 5D AA A0	*A^E@]**
88: A0 C1 D0 D0 CC C5 A0 C3	APPLE C
90: CF CD D0 D5 D4 C5 D2 A0	OMPUTER
98: C9 CE C3 AE A0 A0 AA 01	INC. *A
A0: 1E 06 00 5D AA A0 A0 A0	^F@]**
AB: A0 A0 A0 A0 A0 A0 A0 A0	
B0: A0 A0 A0 A0 A0 A0 A0 A0	
BB: A0 A0 A0 A0 AA 01 1E 07	*A^G
C0: 00 5D AA AA AA AA AA AA	@]*******
C8: AA AA AA AA AA AA AA AA	*****
D0: AA AA AA AA AA AA AA AA	*****
D8: AA AA AA 01 2C 64 00 C3	***A,d@C
E0: D5 D2 71 B1 01 00 03 5D	URq1A@C]
E8: C8 C5 D2 C5 A0 C9 D3 A0	HERE IS
F0: D7 C8 C5 D2 C5 A0 D0 D2	WHERE FR
F8: CF C7 D2 C1 CD A0 C9 D3	OGRAM IS

Figuur 7

#3D0LLL

03D0-	4C BF 9D	JMP	\$9DBF	Jump naar DOS warmstart. Programma intact.
03D3-	4C B4 9D	JMP	\$9DB4	Jump naar DOS coldstart. Reset Basic.
03D6-	4C FD AA	JMP	\$AAFD	Jump naar File Manager.
03D9-	4C B5 B7	JMP	\$B7B5	Jump naar RWTS.
03DC-	AD 0F 9D	LDA	\$9D0F	Korte routine, die gebruikt kan worden
03DF-	AC 0E 9D	LDY	\$9D0E	door machinetaalprogramma. Set input
03E2-	60	RTS		parameters voor aanroep File Manager.
03E3-	AD C2 AA	LDA	\$AAC2	Idem maar voor aanroep RWTS.
03E6-	AC C1 AA	LDY	\$AAC1	
03E9-	60	RTS		
03EA-	4C 51 AB	JMP	\$A851	Jump voor reconnecten DOS intercept
03ED-	EA	NOP		aan keyboard en CRT.
03EE-	EA	NOP		
03EF-	4C 59 FA	JMP	\$FA59	Jump naar BRK afhandelingsroutine.
03F2-	BF	???		Actief bij Autostart ROM.
03F3-	9D 38 4C	STA	\$4C38,X	Herken 9DBF! 3F4 : Power-up byte.
03F6-	58	CLI		3F5 : Jump naar &-routine.
03F7-	FF	???		
03F8-	4C 65 FF	JMP	\$FF65	3F8 : Jump naar CTRL-Y routine.
03FB-	4C 65 FF	JMP	\$FF65	
03FE-	65 FF	ADC	\$FF	3FE : LO/HI adres routine afhandeling
0400-	AA	TAX		maskable interrupt.

De hiervoor opgesomde adressen worden meestal "vectors" genoemd. Het zijn vectors in page 3 (\$300-\$3FF) van de geheugenmap van de APPLE microcomputer, die verwijzen naar DOS-routines en enkele hulproutines.

De reden, dat het DOS aansluitingen vindt in page 3, is dat het DOS op verschillende plaatsen in het geheugen kan resideren. Denk maar aan de verschillende geheugenomvang van een 32 K en een 48 K APPLE computer. Door een aantal standaard-vectors te fixeren, worden moeilijkheden voorkomen met verschillende routine-plaatsen. Het is dan ook begrijpelijk, dat de in page 3 gelegen DOS-vectors vaak worden gebruikt voor DOS-"beveiligingen".

Wanneer je het DOS geladen hebt, kun je de hiervoor afgebeelde geheugenlist oproepen. Tik CALL-151 waarna de prompt verandert in het * -teken. Tik vervolgens 3D0L (return).

Voor degenen, die geen integer Basic hadden geladen, is het spijtig, dat zij de mini-assembler interpretatie moeten missen.

De vector \$3F5 is een bijzonder aantrekkelijke vector, die (mis-) gebruikt kan worden voor eigen machinetaalroutines of Jumps naar het DOS. Test de volgende mogelijkheid:

Verander \$3F6: 98 AD (let op de spatie tussen 98 en AD)-return.

Keer terug naar Basic (zie \$3D0)

en tik: &.

Wat gebeurt er ? U krijgt een CATALOG van uw diskette!

VTOC

00:	04	11	0F	03	00	00	FE	00	D00C00~0
08:	00	00	00	00	00	00	00	00	00000000
10:	00	00	00	00	00	00	00	00	00000000
18:	00	00	00	00	00	00	00	00	00000000
20:	00	00	00	00	00	00	00	7A	0000000z
28:	00	00	00	00	00	00	00	00	00000000
30:	0D	FF	00	00	23	10	00	01	M00#F0A
38:	00	00	00	00	00	00	00	00	00000000
40:	00	00	00	00	00	00	00	00	00000000
48:	00	00	00	00	00	00	00	00	00000000
50:	00	00	00	00	00	00	00	00	00000000
58:	00	00	00	00	00	0F	00	00	00000000
60:	FF	FF	00	00	00	00	00	00	000000
68:	00	7F	00	00	01	FF	00	00	0000A00
70:	00	00	00	00	00	00	00	00	00000000
78:	00	00	00	00	00	00	00	00	00000000
80:	FF	E0	00	00	00	00	00	00	'000000
88:	00	00	00	00	00	00	00	00	00000000
90:	00	00	00	00	00	00	00	00	00000000
98:	00	00	00	00	00	00	00	00	00000000
A0:	00	00	00	00	00	03	00	00	000000C00
AB:	00	00	00	00	00	00	00	00	00000000
B0:	00	00	00	00	00	00	00	00	00000000
B8:	00	00	00	00	00	00	00	00	00000000
C0:	00	00	00	00	00	00	00	00	00000000
CB:	00	00	00	00	00	00	00	00	00000000
D0:	00	00	00	00	00	00	00	00	00000000
DB:	00	00	00	00	00	00	00	00	00000000
E0:	00	00	00	00	00	00	00	00	00000000
EB:	00	00	00	00	00	00	00	00	00000000
F0:	00	00	00	00	00	00	00	00	00000000
FB:	00	00	00	00	00	00	00	00	00000000

Figuur 1

DIRECTORY

00: 00 11 0E 00 00 00 00 00	00N00000
08: 00 00 00 13 0F 82 C8 C5	000SOBHE
10: CC CC CF A0 A0 A0 A0 A0	LLO
18: A0 A0 A0 A0 A0 A0 A0 A0	
20: A0 A0 A0 A0 A0 A0 A0 A0	
28: A0 A0 A0 A0 06 00 14 0F	F0TO
30: 81 C1 CE C9 CD C1 CC D3	AANIMALS
38: A0 A0 A0 A0 A0 A0 A0 A0	
40: A0 A0 A0 A0 A0 A0 A0 A0	
48: A0 A0 A0 A0 A0 A0 A0 12	R
50: 00 15 0F 80 C1 D0 D0 CC	0000APPL
58: C5 A0 D0 D2 CF CD D3 A0	E PROMS
60: A0 A0 A0 A0 A0 A0 A0 A0	
68: A0 A0 A0 A0 A0 A0 A0 A0	
70: A0 A0 03 00 16 0F 81 C1	C0VDAA
78: D0 D0 CC C5 D3 CF C6 D4	PPLESOFT
80: A0 A0 A0 A0 A0 A0 A0 A0	
88: A0 A0 A0 A0 A0 A0 A0 A0	
90: A0 A0 A0 A0 A0 06 00 17	F0W
98: 0F 81 C1 D0 D0 CC C5 D6	0AAPPLEV
A0: C9 D3 C9 CF CE A0 A0 A0	ISION
A8: A0 A0 A0 A0 A0 A0 A0 A0	
B0: A0 A0 A0 A0 A0 A0 A0 A0	
B8: 1A 00 18 0F 81 C2 C9 CF	Z0XDABIO
C0: D2 C8 D9 D4 C8 CD A0 A0	RHYTHM
C8: A0 A0 A0 A0 A0 A0 A0 A0	
D0: A0 A0 A0 A0 A0 A0 A0 A0	
D8: A0 A0 A0 11 00 19 0F 84	00YOD
E0: C2 CF CF D4 B1 B3 A0 A0	BOOT13
E8: A0 A0 A0 A0 A0 A0 A0 A0	
F0: A0 A0 A0 A0 A0 A0 A0 A0	
F8: A0 A0 A0 A0 A0 A0 0A 00	J0

Figuur 2

LL

9D84-	AD E9 B7	LDA	\$B7E9	DOS Coldstart adres *****
9D87-	4A	LSR		
9D88-	4A	LSR		
9D89-	4A	LSR		
9D8A-	4A	LSR		IOBSlot nummer
9D8B-	8D 6A AA	STA	\$AA6A	in variabelen tabel
9D8E-	AD EA B7	LDA	\$B7EA	Drive nummer
9D91-	8D 68 AA	STA	\$AA68	in variabelen tabel
9D94-	AD 00 E0	LDA	\$E000	Controleer welk Basic dialect
9D97-	49 20	EDR	\$\$20	aktief is
9D99-	D0 11	BNE	\$9DAC	Branch naar Applesoft
9D9B-	8D B6 AA	STA	\$AAB6	Integer Basic aanwezig
9D9E-	A2 0A	LDX	\$\$0A	
9DA0-	BD 61 9D	LDA	\$9D61,X	stel de vereiste adressen in
9DA3-	9D 55 9D	STA	\$9D55,X	
9DA6-	CA	DEX		
9DA7-	D0 F7	BNE	\$9DA0	
9DA9-	4C BC 9D	JMP	\$9DBC	
9DAC-	A9 40	LDA	\$\$40	Applesoft Basic aanwezig
9DAE-	8D B6 AA	STA	\$AAB6	
9DB1-	A2 0C	LDX	\$\$0C	stel de vereiste adressen in
9DB3-	BD 68 9D	LDA	\$9D68,X	
9DB6-	9D 55 9D	STA	\$9D55,X	
9DB9-	CA	DEX		
9DBA-	D0 F7	BNE	\$9DB3	
9DBC-	38	SEC		
9DBD-	B0 12	BCS	\$9DD1	
9DBF-	AD B6 AA	LDA	\$AAB6	DOS Warmstart adres *****
9DC2-	D0 04	BNE	\$9DC8	
9DC4-	A9 20	LDA	\$\$20	Stel Integer Basic in
9DC6-	D0 05	BNE	\$9DCD	
9DC8-	0A	ASL		
9DC9-	10 05	BPL	\$9DD0	
9DCB-	A9 4C	LDA	\$\$4C	Stel Applesoft Basic in
9DCD-	20 B2 A5	JSR	\$A5B2	
9DD0-	18	CLC		
9DD1-	08	PHP		Initialiseer DOS
9DD2-	20 51 AB	JSR	\$A851	
9DD5-	A9 00	LDA	\$\$00	Nomon C,I,0
9DD7-	8D 5E AA	STA	\$AA5E	
9DDA-	8D 52 AA	STA	\$AA52	Initialiseer State 0
9DDD-	28	PLP		
9DDE-	6A	ROR		
9DDF-	8D 51 AA	STA	\$AA51	
9DE2-	30 03	BMI	\$9DE7	
9DE4-	6C 5E 9D	JMP	(\$9D5E)	
9DE7-	6C 5C 9D	JMP	(\$9D5C)	
9DEA-	0A	ASL		Controleer RAM Applesoft
9DEB-	10 19	BPL	\$9E06	
9DED-	8D B6 AA	STA	\$AAB6	Indien zo, dat pointers op hun plaats zetten
9DF0-	A2 0C	LDX	\$\$0C	
9DF2-	BD 77 9D	LDA	\$9D77,X	
9DF5-	9D 55 9D	STA	\$9D55,X	
9DF8-	CA	DEX		
9DF9-	D0 F7	BNE	\$9DF2	
9DFB-	A2 1D	LDX	\$\$1D	
9DFD-	BD 93 AA	LDA	\$AA93,X	
9E00-	9D 75 AA	STA	\$AA75,X	
9E03-	CA	DEX		
9E04-	10 F7	BPL	\$9DFD	

9E06-	AD B1 AA	LDA	\$AAB1	Initialiseer MAXFILES (vervalt naar 3)
9E09-	8D 57 AA	STA	\$AA57	
9E0C-	20 D4 A7	JSR	\$A7D4	Set up buffers
9E0F-	AD B3 AA	LDA	\$AAB3	EXEC controle - wordt na volgende karakter
9E12-	F0 09	BEQ	\$9E1D	op non-actief gesteld.
9E14-	48	PHA		
9E15-	20 9D A6	JSR	\$A69D	
9E18-	68	PLA		
9E19-	A0 00	LDY	#\$00	
9E1B-	91 40	STA	(\$40),Y	
9E1D-	20 5B A7	JSR	\$A75B	Reset CSW state op 0
9E20-	AD 5F AA	LDA	\$AA5F	is dus warmstart status
9E23-	D0 20	BNE	\$9E45	Zitten we midden in commando ?
9E25-	A2 2F	LDX	#\$2F	Nee, dan \$3D0 instellen
9E27-	8D 51 9E	LDA	\$9E51,X	Haal lijst op
9E2A-	9D D0 03	STA	\$03D0,X	en zet hem weg!
9E2D-	CA	DEX		
9E2E-	10 F7	BPL	\$9E27	
9E30-	AD 53 9E	LDA	\$9E53	
9E33-	8D F3 03	STA	\$03F3	
9E36-	49 A5	EOR	#\$A5	
9E38-	8D F4 03	STA	\$03F4	
9E3B-	AD 52 9E	LDA	\$9E52	
9E3E-	8D F2 03	STA	\$03F2	
9E41-	A9 06	LDA	#\$06	
9E43-	D0 05	BNE	\$9E4A	
9E45-	AD 62 AA	LDA	\$AA62	Test lopend commando
9E48-	F0 06	BEQ	\$9E50	anders terug naar Basic
9E4A-	8D 5F AA	STA	\$AA5F	
9E4D-	4C 80 A1	JMP	\$A180	
9E50-	60	RTS		Klaar
9E51-	4C BF 9D	JMP	\$9DBF	De lijst schuift naar \$3D0 *****
9E54-	4C 84 9D	JMP	\$9DB4	
9E57-	4C FD AA	JMP	\$AAF0	
9E5A-	4C B5 B7	JMP	\$B7B5	
9E5D-	AD 0F 9D	LDA	\$9D0F	
9E60-	AC 0E 9D	LDY	\$9D0E	
9E63-	60	RTS		
9E64-	AD C2 AA	LDA	\$AAC2	
9E67-	AC C1 AA	LDY	\$AAC1	
9E6A-	60	RTS		
9E6B-	4C 51 A8	JMP	\$A851	
9E6E-	EA	NOP		
9E6F-	EA	NOP		
9E70-	4C 59 FA	JMP	\$FA59	
9E73-	4C 65 FF	JMP	\$FF65	
9E76-	4C 58 FF	JMP	\$FF58	
9E79-	4C 65 FF	JMP	\$FF65	
9E7C-	4C 65 FF	JMP	\$FF65	
9E7F-	65 FF	ADC	\$FF	
9E81-	20 D1 9E	JSR	\$9ED1	Keyboard intercept routine
9E84-	AD 51 AA	LDA	\$AA51	wordt er een file gelezen ?
9E87-	F0 15	BEQ	\$9E9E	
9E89-	48	PHA		Indien ja, dan cursor weg
9E8A-	AD 5C AA	LDA	\$AA5C	
9E8D-	91 28	STA	(\$28),Y	
9E8F-	68	PLA		
9E90-	30 03	BMI	\$9E95	Is dit een Coldstart ?
9E92-	4C 26 A6	JMP	\$A626	Nee, ik lees een byte uit een file!
9E95-	20 EA 9D	JSR	\$9DEA	Ja, dan eerst registers goedzetten
9E98-	A4 24	LDY	\$24	
9E9A-	A9 60	LDA	#\$60	

9E9C-	91 28	STA	(\$2B),Y	
9E9E-	AD B3 AA	LDA	\$AAB3	Bezig met EXEC ?
9EA1-	F0 03	BEQ	\$9EA6	
9EA3-	20 B2 A6	JSR	\$A6B2	Haal dan volgende byte uit EXEC file
9EA6-	A9 03	LDA	##03	
9EAB-	BD 52 AA	STA	\$AA52	
9EAB-	20 BA 9F	JSR	\$9FBA	Restore de registers t.b.v. DOS
9EAE-	20 BA 9E	JSR	\$9EBA	Naar input routine
9EB1-	BD 5C AA	STA	\$AA5C	Save inputkarakter
9EB4-	BE 5A AA	STX	\$AA5A	Zet X weg met nieuwe waarde
9EB7-	4C B3 9F	JMP	\$9FB3	DOS exit
9EBA-	6C 38 00	JMP	(\$003B)	Spring naar KSW routine
9EBD-	20 D1 9E	JSR	\$9ED1	Save registers t.b.v. DOS
9ECO-	AD 52 AA	LDA	\$AA52	Haal de video intercept pointer op
9EC3-	0A	ASL		en gebruik hem als index
9EC4-	AA	TAX		voor de State Handler tabel
9EC5-	BD 11 9D	LDA	\$9D11,X	vanaf 9D10 . Niet opgenomen in listing
9EC8-	4B	PHA		
9EC9-	BD 10 9D	LDA	\$9D10,X	Bepaal nu de funktie en spring er
9ECC-	4B	PHA		heen.
9ECD-	AD 5C AA	LDA	\$AA5C	Save A register
9EDO-	60	RTS		Klaar ---
9ED1-	BD 5C AA	STA	\$AA5C	Register opberg routine *****
9ED4-	BE 5A AA	STX	\$AA5A	
9ED7-	BC 5B AA	STY	\$AA5B	
9EDA-	BA	TSX		
9EDB-	EB	INX		
9EDC-	EB	INX		
9EDD-	BE 59 AA	STX	\$AA59	
9EE0-	A2 03	LDX	##03	
9EE2-	BD 53 AA	LDA	\$AA53,X	Stel de I/O afhandeling in
9EE5-	95 36	STA	\$36,X	voor KSW en CSW vectors
9EE7-	CA	DEX		van \$36-\$39
9EE8-	10 F8	BPL	\$9EE2	
9EEA-	60	RTS		Klaar ---
9EEB-	AE B7 AA	LDX	\$AAB7	State 0 output afhandeling *****
9EEE-	F0 03	BEQ	\$9EF3	Begin van een programma/commandoregel
9EF0-	4C 78 9F	JMP	\$9F78	Bezig met "RUN" ? Verder dan!
9EF3-	AE 51 AA	LDX	\$AA51	Of zijn we bezig met een file ?
9EF6-	F0 08	BEQ	\$9F00	Als de read-flag er is, kijk dan naar
9EF8-	C9 BF	CMP	##BF	"?" als code voor Basic input
9EFA-	F0 75	BEQ	\$9F71	Naar State 6 en laat hem weg.
9EFC-	C5 33	CMP	\$33	Is het het begin van een regel ?
9EFE-	F0 27	BEQ	\$9F27	Dan naar State 2 voor afhandeling
9F00-	A2 02	LDX	##02	
9F02-	BE 52 AA	STX	\$AA52	
9F05-	CD B2 AA	CMP	\$AAB2	Als het eerste karakter geen CTRL-D
9F08-	D0 19	BNE	\$9F23	is, dan doorgaan met State 2
9F0A-	CA	DEX		
9F0B-	BE 52 AA	STX	\$AA52	
9F0E-	CA	DEX		
9F0F-	BE 5D AA	STX	\$AA5D	
9F12-	AE 5D AA	LDX	\$AA5D	State 1 output afhandeling *****
9F15-	9D 00 02	STA	\$0200,X	Berg inputkarakter in input-buffer op
9F18-	EB	INX		
9F19-	BE 5D AA	STX	\$AA5D	
9F1C-	C9 8D	CMP	##8D	Zie je deze "return"?
9F1E-	D0 75	BNE	\$9F95	Nee, dan eindigen via 9F95
9F20-	4C CD 9F	JMP	\$9FCD	Anders naar command scanner
9F23-	C9 8D	CMP	##8D	State 2 output afhandeling *****
9F25-	D0 7D	BNE	\$9FA4	Echo karakter op het scherm als het
9F27-	A2 00	LDX	##00	geen "return" is. Anders State 0 doen.

9F29-	BE 52 AA	STX	\$AA52	Berg de CSW state op
9F2C-	4C A4 9F	JMP	\$9FA4	Jump naar 'echo karakters op scherm'
9F2F-	A2 00	LDX	#\$00	State 3 output afhandeling *****
9F31-	BE 52 AA	STX	\$AA52	De commando status wordt 0 {input}
9F34-	C9 8D	CMP	#\$8D	Einde van de regel ?
9F36-	F0 07	BEQ	\$9F3F	Naar afhandeling EOL/einde regel
9F38-	AD B3 AA	LDA	\$AAB3	Controleer EXEC flag
9F3B-	F0 67	BEQ	\$9FA4	Ga naar 'echo karakter'
9F3D-	D0 5E	BNE	\$9F9D	Ga naar 'MON I'
9F3F-	48	PHA		Aha, einde van de regel!
9F40-	38	SEC		Als er geen Basic- of EXEC programma
9F41-	AD B3 AA	LDA	\$AAB3	bezig is (RUN), bekijk dan of er
9F44-	D0 03	BNE	\$9F49	een DOS commando komt.
9F46-	20 5E A6	JSR	\$A65E	
9F49-	68	PLA		
9F4A-	90 EC	BCC	\$9F38	
9F4C-	AE 5A AA	LDX	\$AA5A	Anders wordt het karakter weergegeven
9F4F-	4C 15 9F	JMP	\$9F15	(geëchood) en switch naar State 1
9F52-	C9 8D	CMP	#\$8D	State 4 output afhandeling *****
9F54-	D0 05	BNE	\$9F5B	"return" ? {write to file}
9F56-	A9 05	LDA	#\$05	
9F58-	BD 52 AA	STA	\$AA52	CSW State
9F5B-	20 0E A6	JSR	\$A60E	Write byte naar disk file
9F5E-	4C 99 9F	JMP	\$9F99	Exit DOS indien MON 0.
9F61-	CD B2 AA	CMP	\$AAB2	State 5 output afhandeling *****
9F64-	F0 85	BEQ	\$9EEB	Bij CTRL-D naar State 0 {write data}
9F66-	C9 8A	CMP	#\$8A	Als er een line-feed is,
9F68-	F0 F1	BEQ	\$9F5B	gewoon regel afmaken
9F6A-	A2 04	LDX	#\$04	Anders terug naar State 4
9F6C-	BE 52 AA	STX	\$AA52	
9F6F-	D0 E1	BNE	\$9F52	
9F71-	A9 00	LDA	#\$00	State 6 output afhandeling *****
9F73-	BD 52 AA	STA	\$AA52	Set CSW State op 0 {prompt weg}
9F76-	F0 25	BEQ	\$9F9D	Echo karakter bij MON I
9F78-	A9 00	LDA	#\$00	Afloop van het RUN commando
9F7A-	BD B7 AA	STA	\$AAB7	Even bijhouden
9F7D-	20 51 A8	JSR	\$AB51	Keyboard vectors op hun plaatsen
9F80-	4C DC A4	JMP	\$A4DC	en de routine afsluiten
9F83-	AD 00 02	LDA	\$0200	Einde commando scan via buffer
9F86-	CD B2 AA	CMP	\$AAB2	CTRL-D ? Dan echo/weergeven
9F89-	F0 0A	BEQ	\$9F95	naar 'echo karakter' en exit DOS
9F8B-	A9 8D	LDA	#\$8D	Doe anders net of er een return was
9F8D-	BD 00 02	STA	\$0200	en zorg dat het DOS dit ziet.
9F90-	A2 00	LDX	#\$00	
9F92-	BE 5A AA	STX	\$AA5A	
9F95-	A9 40	LDA	#\$40	Echo karakter routines en exit DOS ***
9F97-	D0 06	BNE	\$9F9F	Echo indien MON C
9F99-	A9 10	LDA	#\$10	of vervolg
9F9B-	D0 02	BNE	\$9F9F	Echo indien MON 0
9F9D-	A9 20	LDA	#\$20	Echo indien MON I
9F9F-	2D 5E AA	AND	\$AA5E	AND MON flags
9FA2-	F0 0F	BEQ	\$9FB3	DOS exit routine pakken
9FA4-	20 BA 9F	JSR	\$9FBA	Herstel de registers t.b.v. het DOS
9FA7-	20 C5 9F	JSR	\$9FC5	Echo het karakter naar het scherm
9FAA-	BD 5C AA	STA	\$AA5C	Save A register
9FAD-	BC 5B AA	STY	\$AA5B	Save Y register
9FBO-	BE 5A AA	STX	\$AA5A	Save X register
9FB3-	20 51 A8	JSR	\$AB51	Herstel registers
9FB6-	AE 59 AA	LDX	\$AA59	
9FB9-	9A	TXS		
9FBA-	AD 5C AA	LDA	\$AA5C	
9FBD-	AC 5B AA	LDY	\$AA5B	

9FC0-	AE 5A AA	LDX	\$AA5A	
9FC3-	38	SEC		
9FC4-	60	RTS		einde DOS operatie
9FC5-	6C 36 00	JMP	(\$0036)	Jump naar ware CSW routine \$36-37)
9FC8-	A9 8D	LDA	#\$8D	return
9FCA-	4C C5 9F	JMP	\$9FC5	Jump naar COUT
9FCD-	A0 FF	LDY	#\$FF	DOS commando routine *****
9FCF-	8C 5F AA	STY	\$AA5F	Ø in Y van commando index tabel
9FD2-	C8	INY		en reset de index voor
9FD3-	8C 62 AA	STY	\$AA62	"commando's in uitvoering"
9FD6-	EE 5F AA	INC	\$AA5F	Schuif commando index op
9FD9-	A2 00	LDX	#\$00	
9FDB-	08	PHP		haal karakters uit inputbuffer
9FDC-	BD 00 02	LDA	\$0200,X	we willen geen CTRL-D meer zien
9FDF-	CD B2 AA	CMP	\$AAB2	dus afhandelen
9FE2-	D0 01	BNE	\$9FE5	of indien afwezig verder lezen
9FE4-	E8	INX		
9FE5-	8E 5D AA	STX	\$AA5D	X als commando teller
9FEB-	20 A4 A1	JSR	\$A1A4	Subroutine die spaties weghaalt
9FEB-	29 7F	AND	#\$7F	gevonden karakter(s) aanpassen
9FED-	59 84 AB	EOR	\$AB84,Y	en vergelijken met commandotabel
9FF0-	C8	INY		en als het nog niet klopt, kijkt
9FF1-	0A	ASL		dan of er meer karakters volgen
9FF2-	F0 02	BEQ	\$9FF6	
9FF4-	68	PLA		gooi de spaties eruit
9FF5-	08	PHP		
9FF6-	70 F0	BCC	\$9FE8	en loop alles opnieuw na!
9FF8-	28	PLP		
9FF9-	F0 20	BEQ	\$A01B	bereken nieuwe index
9FFB-	B9 84 AB	LDA	\$AB84,Y	voor de commando-herkenning
9FFE-	D0 D6	BNE	\$9FD6	keer terug naar 9FD6
A000-	AD 00 02	LDA	\$0200	Geen commando ?
A003-	CD B2 AA	CMP	\$AAB2	Toch geen CTRL-D ?
A006-	F0 03	BEQ	\$A00B	
A008-	4C A4 9F	JMP	\$9FA4	Werk registers bij
A00B-	AD 01 02	LDA	\$0201	volgend karakter
A00E-	C9 8D	CMP	#\$8D	Aha, return ?
A010-	D0 06	BNE	\$A018	Nee, dan is het goed
A012-	20 5B A7	JSR	\$A75B	want anders gaat er
A015-	4C 95 9F	JMP	\$9F95	nu direkt een foutmelding komen
A018-	4C C4 A6	JMP	\$A6C4	"Syntax Error"!
A01B-	0E 5F AA	ASL	\$AA5F	Bereken een index voor de operand
A01E-	AC 5F AA	LDY	\$AA5F	tabel van gevonden commando's
A021-	20 5E A6	JSR	\$A65E	Is Basic operationeel ?
A024-	90 0C	BCC	\$A032	Anders doorschuiven naar A032
A026-	A9 02	LDA	#\$02	Indien Basic niet operationeel is
A028-	39 09 A9	AND	\$A909,Y	en het niet gaat om een "direct"
A02B-	F0 05	BEQ	\$A032	commando dan wordt het spannend!
A02D-	A9 0F	LDA	#\$0F	Even instellen:
A02F-	4C D2 A6	JMP	\$A6D2	"NOT DIRECT COMMAND"
A032-	C0 06	CPY	#\$06	
A034-	D0 02	BNE	\$A038	Nu wordt de prompt weggehaald
A036-	84 33	STY	\$33	
A038-	A9 20	LDA	#\$20	Is een filenaam vereist ?
A03A-	39 09 A9	AND	\$A909,Y	
A03D-	F0 61	BEQ	\$A0A0	Geen naam te vinden! Naar \$A0A0
A03F-	20 95 A0	JSR	\$A095	Clear de filenaam buffer
A042-	08	PHP		
A043-	20 A4 A1	JSR	\$A1A4	Weg met spaties
A046-	F0 1E	BEQ	\$A066	
A04B-	0A	ASL		Vergelijk nu de filenaam
A049-	90 05	BCC	\$A050	met de filenaam in de Disk-buffer

A04B-	30 03	BMI	\$A050	
A04D-	4C 00 A0	JMP	\$A000	
A050-	6A	RDR		
A051-	4C 59 A0	JMP	\$A059	Filenaam afhandeling
A054-	20 93 A1	JSR	\$A193	
A057-	F0 0D	BEQ	\$A066	
A059-	99 75 AA	STA	\$AA75,Y	
A05C-	CB	INY		
A05D-	C0 3C	CPY	##3C	Kijk of er een ", " volgt!
A05F-	90 F3	BCC	\$A054	
A061-	20 93 A1	JSR	\$A193	
A064-	D0 FB	BNE	\$A061	
A066-	2B	PLP		
A067-	D0 0F	BNE	\$A07B	Het kan ook zijn, dat er een
A069-	AC 5F AA	LDY	\$AA5F	filenaam nodig is, waar er
A06C-	A9 10	LDA	##10	geen werd opgegeven.
A06E-	39 09 A9	AND	\$A909,Y	Ofdat een commando geen filenaam
A071-	F0 0C	BEQ	\$A07F	vereist: b.v. SAVE →tape
A073-	A0 1E	LDY	##1E	
A075-	0B	PHP		
A076-	D0 CB	BNE	\$A043	
A07B-	AD 93 AA	LDA	\$AA93	
A07B-	C9 A0	CMP	##A0	Anders zal er straks een
A07D-	F0 13	BEQ	\$A092	foutmelding volgen.
A07F-	AD 75 AA	LDA	\$AA75	
A082-	C9 A0	CMP	##A0	
A084-	D0 4B	BNE	\$A0D1	
A086-	AC 5F AA	LDY	\$AA5F	
A089-	A9 C0	LDA	##C0	
A08B-	39 09 A9	AND	\$A909,Y	
A08E-	F0 02	BEQ	\$A092	
A090-	10 3F	BPL	\$A0D1	
A092-	4C 00 A0	JMP	\$A000	
A095-	A0 3C	LDY	##3C	Subroutine voor het opheffen
A097-	A9 A0	LDA	##A0	van de filenamen
A099-	99 74 AA	STA	\$AA74,Y	
A09C-	BB	DEY		
A09D-	D0 FA	BNE	\$A099	
A09F-	60	RTS		Klaar ---
A0A0-	BD 75 AA	STA	\$AA75	Er was geen filenaam vereist
A0A3-	A9 0C	LDA	##0C	Was er een positionele operand
A0A5-	39 09 A9	AND	\$A909,Y	vereist ?
A0AB-	F0 27	BEQ	\$A0D1	Nee, dan naar A0D1
A0AA-	20 B9 A1	JSR	\$A1B9	Anders moet deze worden geconverteerd
A0AD-	B0 1F	BCS	\$A0CE	Of werd het getal weggelaten ?
A0AF-	AB	TAY		
A0B0-	D0 17	BNE	\$A0C9	
A0B2-	E0 11	CPX	##11	Was het getal binnen de
A0B4-	B0 13	BCS	\$A0C9	gestelde grenzen ?
A0B6-	AC 5F AA	LDY	\$AA5F	
A0B9-	A9 0B	LDA	##0B	
A0BB-	39 09 A9	AND	\$A909,Y	Controleer het
A0BE-	F0 06	BEQ	\$A0C6	
A0C0-	E0 0B	CPX	##0B	
A0C2-	B0 CE	BCS	\$A092	
A0C4-	90 0B	BCC	\$A0D1	Dik voor mekaar!
A0C6-	BA	TXA		Want anders valt
A0C7-	D0 0B	BNE	\$A0D1	er een foutmelding te
A0C9-	A9 02	LDA	##02	verwachten:
A0CB-	4C D2 A6	JMP	\$A6D2	"RANGE ERROR"
A0CE-	4C C4 A6	JMP	\$A6C4	
A0D1-	A9 00	LDA	##00	Set vervalwaarden (V=0,B=0,L=0)

A0D3-	BD 65 AA	STA	\$AA65	Keyword dat werd gevonden
A0D6-	BD 74 AA	STA	\$AA74	Mon
A0D9-	BD 66 AA	STA	\$AA66	Vol
A0DC-	BD 6C AA	STA	\$AA6C	Len L
A0DF-	BD 6D AA	STA	\$AA6D	Len H
A0E2-	20 DC BF	JSR	\$BFDC	Byte
A0E5-	AD 5D AA	LDA	\$AA5D	Index
A0E8-	20 A4 A1	JSR	\$A1A4	Routine t.b.v. positionele operand
A0EB-	D0 1F	BNE	\$A10C	onder weglating van spaties
A0ED-	C9 BD	CMP	#\$BD	return gevonden ?
A0EF-	D0 F7	BNE	\$A0E8	anders zijn we nog niet klaar!
A0F1-	AE 5F AA	LDX	\$AA5F	Command index in X
A0F4-	AD 65 AA	LDA	\$AA65	Keyword index in A
A0F7-	1D 0A A9	ORA	\$A90A,X	Hebben we nu het hele commando
A0FA-	5D 0A A9	EOR	\$A90A,X	binnen gehaald ?
A0FD-	D0 93	BNE	\$A092	
A0FF-	AE 63 AA	LDX	\$AA63	
A102-	F0 76	BEQ	\$A17A	OK, verwerk dan nu het commando.
A104-	BD 63 AA	STA	\$AA63	
A107-	BE 5D AA	STX	\$AA5D	
A10A-	D0 DC	BNE	\$A0E8	
A10C-	A2 0A	LDX	#\$0A	Dan gaan we nu het opgegeven
A10E-	DD 40 A9	CMP	\$A940,X	commando vergelijken met de DOS tabel
A111-	F0 05	BEQ	\$A118	Als het niet klopt
A113-	CA	DEX		volgt er "Syntax Error"
A114-	D0 F8	BNE	\$A10E	
A116-	F0 B6	BEQ	\$A0CE	
A118-	BD 4A A9	LDA	\$A94A,X	Afhandeling positionele operand
A11B-	30 47	BMI	\$A164	
A11D-	0D 65 AA	ORA	\$AA65	
A120-	BD 65 AA	STA	\$AA65	
A123-	CA	DEX		
A124-	BE 64 AA	STX	\$AA64	
A127-	20 B9 A1	JSR	\$A1B9	Haal waarde op van keyword
A12A-	B0 A2	BCS	\$A0CE	
A12C-	AD 64 AA	LDA	\$AA64	Controle de waarde
A12F-	0A	ASL		
A130-	0A	ASL		
A131-	AB	TAY		aan de hand
A132-	A5 45	LDA	\$45	
A134-	D0 09	BNE	\$A13F	van de
A136-	A5 44	LDA	\$44	
A138-	D9 55 A9	CMP	\$A955,Y	Waarden tabel
A13B-	90 BC	BCC	\$A0C9	
A13D-	A5 45	LDA	\$45	
A13F-	D9 58 A9	CMP	\$A958,Y	
A142-	90 08	BCC	\$A14F	
A144-	D0 83	BNE	\$A0C9	
A146-	A5 44	LDA	\$44	
A148-	D9 57 A9	CMP	\$A957,Y	
A14B-	90 02	BCC	\$A14F	
A14D-	D0 F5	BNE	\$A144	
A14F-	AD 63 AA	LDA	\$AA63	
A152-	D0 94	BNE	\$A0E8	
A154-	98	TYA		
A155-	4A	LSR		
A156-	AB	TAY		
A157-	A5 45	LDA	\$45	
A159-	99 67 AA	STA	\$AA67,Y	Bewaar de waarde van het keyword
A15C-	A5 44	LDA	\$44	
A15E-	99 66 AA	STA	\$AA66,Y	
A161-	4C EB A0	JMP	\$A0E8	

A164-	48	PHA		
A165-	A9 80	LDA	##80	Set de juiste bits, indien
A167-	0D 65 AA	ORA	##AA65	MON of NOMON werd ontdekt.
A16A-	8D 65 AA	STA	##AA65	
A16D-	68	PLA		
A16E-	29 7F	AND	##7F	
A170-	0D 74 AA	ORA	##AA74	
A173-	8D 74 AA	STA	##AA74	
A176-	D0 E9	BNE	##A161	
A178-	F0 9C	BEQ	##A116	
A17A-	20 80 A1	JSR	##A180	Proces geldig DOS commando
A17D-	4C 83 9F	JMP	##9F83	Echo via 9F83
A180-	20 5B A7	JSR	##A75B	D0 Command *****
A183-	20 AE A1	JSR	##A1AE	Clear file manager parm list
A186-	AD 5F AA	LDA	##AA5F	Haal commando index op
A189-	AA	TAX		
A18A-	BD 1F 9D	LDA	##9D1F,X	Adres van de betreffende
A18D-	48	PHA		commando afhandelingsroutines
A18E-	BD 1E 9D	LDA	##9D1E,X	
A191-	48	PHA		
A192-	60	RTS		Klaar ----
A193-	AE 5D AA	LDX	##AA5D	Haal volgend karakter van de
A196-	BD 00 02	LDA	##0200,X	commando regel
A199-	C9 8D	CMP	##8D	Was dit een return?
A19B-	F0 06	BEQ	##A1A3	dan klaar
A19D-	E8	INX		
A19E-	8E 5D AA	STX	##AA5D	houdt index bij
A1A1-	C9 AC	CMP	##AC	hé, een ", " ?
A1A3-	60	RTS		Klaar ----
A1A4-	20 93 A1	JSR	##A193	Werk de spaties weg
A1A7-	F0 FA	BEQ	##A1A3	Z-flag is set, indien ", "
A1A9-	C9 A0	CMP	##A0	of return als eerste (niet-spatie)
A1AB-	F0 F7	BEQ	##A1A4	karakter wordt ontmoet.
A1AD-	60	RTS		Klaar ----
A1AE-	A9 00	LDA	##00	File manager parameter lijst
A1B0-	A0 16	LDY	##16	wordt nu op 0 gezet
A1B2-	99 BA B5	STA	##B5BA,Y	
A1B5-	88	DEY		
A1B6-	D0 FA	BNE	##A1B2	
A1B8-	60	RTS		Klaar ----
A1B9-	A9 00	LDA	##00	Converteer het numerieke,
A1BB-	85 44	STA	##44	ASCII gecodeerde karakter
A1BD-	85 45	STA	##45	in het binaire getal
A1BF-	20 A4 A1	JSR	##A1A4	
A1C2-	08	PHP		
A1C3-	C9 A4	CMP	##A4	Compair "\$" teken
A1C5-	F0 3C	BEQ	##A203	Is het een hexadecimale code ?
A1C7-	28	PLP		
A1C8-	4C CE A1	JMP	##A1CE	Nee, een decimale.
A1CB-	20 A4 A1	JSR	##A1A4	Weg met de spaties
A1CE-	D0 06	BNE	##A1D6	Conversieroutine
A1D0-	A6 44	LDX	##44	
A1D2-	A5 45	LDA	##45	
A1D4-	18	CLC		
A1D5-	60	RTS		
A1D6-	38	SEC		
A1D7-	E9 B0	SBC	##B0	
A1D9-	30 21	BMI	##A1FC	
A1DB-	C9 0A	CMP	##0A	
A1DD-	B0 1D	BCS	##A1FC	
A1DF-	20 FE A1	JSR	##A1FE	
A1E2-	65 44	ADC	##44	

A1E4-	AA	TAX		
A1E5-	A9 00	LDA	#\$00	
A1E7-	65 45	ADC	\$45	
A1E9-	A8	TAY		
A1EA-	20 FE A1	JSR	\$A1FE	
A1ED-	20 FE A1	JSR	\$A1FE	
A1F0-	8A	TXA		
A1F1-	65 44	ADC	\$44	
A1F3-	85 44	STA	\$44	
A1F5-	98	TYA		
A1F6-	65 45	ADC	\$45	
A1F8-	85 45	STA	\$45	
A1FA-	90 CF	BCC	\$A1CB	
A1FC-	38	SEC		
A1FD-	60	RTS		Klaar ----
A1FE-	06 44	ASL	\$44	Vermenigvuldig
A200-	26 45	ROL	\$45	
A202-	60	RTS		Decimale conversie klaar---
A203-	28	PLP		
A204-	20 A4 A1	JSR	\$A1A4	weg met de spaties
A207-	F0 C5	BEQ	\$A1CE	Begin van de hex conversie
A209-	38	SEC		
A20A-	E9 B0	SBC	##B0	
A20C-	30 EE	BMI	\$A1FC	
A20E-	C9 0A	CMP	##0A	
A210-	90 08	BCC	\$A21A	
A212-	E9 07	SBC	##07	
A214-	30 E6	BMI	\$A1FC	
A216-	C9 10	CMP	##10	
A218-	B0 E2	BCS	\$A1FC	
A21A-	A2 04	LDX	##04	
A21C-	20 FE A1	JSR	\$A1FE	
A21F-	CA	DEX		
A220-	D0 FA	BNE	\$A21C	
A222-	05 44	ORA	\$44	
A224-	85 44	STA	\$44	
A226-	4C 04 A2	JMP	\$A204	
A229-	A5 44	LDA	\$44	****PR#x afhandeling *****
A22B-	4C 95 FE	JMP	\$FE95	Exit via ROM
A22E-	A5 44	LDA	\$44	****IN#x afhandeling *****
A230-	4C 8B FE	JMP	\$FE8B	Exit via ROM
A233-	AD 5E AA	LDA	\$AA5E	****MON afhandeling *****
A236-	0D 74 AA	ORA	\$AA74	
A239-	8D 5E AA	STA	\$AA5E	Set nieuwe flags
A23C-	60	RTS		
A23D-	2C 74 AA	BIT	\$AA74	****NOMON afhandeling *****
A240-	50 03	BVC	\$A245	
A242-	20 C8 9F	JSR	\$9FC8	Indien C was gegeven,
A245-	A9 70	LDA	##70	geef dan een return
A247-	4D 74 AA	EOR	\$AA74	
A24A-	2D 5E AA	AND	\$AA5E	Set de juiste bits
A24D-	8D 5E AA	STA	\$AA5E	
A250-	60	RTS		Klaar----
A251-	A9 00	LDA	#\$00	MAXFILES afhandeling
A253-	8D B3 AA	STA	\$AAB3	Stop EXEC indien operationeel
A256-	A5 44	LDA	\$44	
A258-	48	PHA		
A259-	20 16 A3	JSR	\$A316	Close files
A25C-	68	PLA		
A25D-	8D 57 AA	STA	\$AA57	Set nieuwe Maxfileswaarde in tabel
A260-	4C D4 A7	JMP	\$A7D4	Bouw nieuwe filebuffers
A263-	A9 05	LDA	##05	DELETE afhandeling

A265-	20 AA A2	JSR	\$A2AA	Ga naar afhandelingsroutine
A268-	20 64 A7	JSR	\$A764	Maak de buffer die door de file werd
A26B-	A0 00	LDY	#\$00	gebruikt weer vrij.
A26D-	98	TYA		
A26E-	91 40	STA	(\$40),Y	
A270-	60	RTS		Klaar----
A271-	A9 07	LDA	#\$07	LOCK/UNLOCK afhandeling *****
A273-	D0 02	BNE	\$A277	Haal lock opcode uit file manager
A275-	A9 08	LDA	#\$08	of de unlock opcode
A277-	20 AA A2	JSR	\$A2AA	Set file manager 'open' voor de functie
A27A-	4C EA A2	JMP	\$A2EA	Keer tenslotte terug via Close
A27D-	A9 0C	LDA	#\$0C	VERIFY afhandeling *****
A27F-	D0 F6	BNE	\$A277	Voer de functie uit
A281-	AD 08 9D	LDA	\$9D08	RENAME afhandeling *****
A284-	8D BD B5	STA	\$B5BD	Berg adres 2e naam op in filemanager
A287-	AD 09 9D	LDA	\$9D09	
A28A-	8D BE B5	STA	\$B5BE	
A28D-	A9 09	LDA	#\$09	Haal de functie opcode op
A28F-	8D 63 AA	STA	\$AA63	en zet hem weg
A292-	20 C8 A2	JSR	\$A2C8	Naar de file manager driver
A295-	4C EA A2	JMP	\$A2EA	en stop via Close
A298-	20 A3 A2	JSR	\$A2A3	APPEND afhandeling *****
A29B-	20 8C A6	JSR	\$A6BC	Open file en lees hem tot 0
A29E-	D0 FB	BNE	\$A29B	wordt ontmoet. Toggel Append flag.
A2A0-	4C 71 B6	JMP	\$B671	Keer terug via POSITION
A2A3-	A9 00	LDA	#\$00	OPEN afhandeling *****
A2A5-	4C D5 A3	JMP	\$A3D5	Ga naar afhandelingsroutine
A2A8-	A9 01	LDA	#\$01	File Manager Controle Routine ****
A2AA-	8D 63 AA	STA	\$AA63	"open" opcode. Indien geen L waarde,
A2AD-	AD 6C AA	LDA	\$AA6C	dan 0001 wegzetten in parmlist
A2B0-	D0 0A	BNE	\$A2BC	
A2B2-	AD 6D AA	LDA	\$AA6D	L adres +1
A2B5-	D0 05	BNE	\$A2BC	
A2B7-	A9 01	LDA	#\$01	
A2B9-	8D 6C AA	STA	\$AA6C	L waarde verschuiven
A2BC-	AD 6C AA	LDA	\$AA6C	
A2BF-	8D BD B5	STA	\$B5BD	record nummer
A2C2-	AD 6D AA	LDA	\$AA6D	
A2C5-	8D BE B5	STA	\$B5BE	recnum +1
A2C8-	20 EA A2	JSR	\$A2EA	Close reeds geopende file.
A2CB-	A5 45	LDA	\$45	
A2CD-	D0 03	BNE	\$A2D2	Is er een geschikte buffer ?
A2CF-	4C C8 A6	JMP	\$A6C8	"NO FILE BUFFER AVAILABLE"!
A2D2-	85 41	STA	\$41	Waar is de buffer ?
A2D4-	A5 44	LDA	\$44	
A2D6-	85 40	STA	\$40	Compleet het adres
A2D8-	20 43 A7	JSR	\$A743	Copiër filenaam in buffer
A2DB-	20 4E A7	JSR	\$A74E	Copiër filepointers in f.m. parmlist
A2DE-	20 1A A7	JSR	\$A71A	Rond die handeling af
A2E1-	AD 63 AA	LDA	\$AA63	Pik variabele uit datalist {index}
A2E4-	8D BB B5	STA	\$B5BB	Zet indexcode van de functie in parmlist
A2E7-	4C AB A6	JMP	\$A6AB	Exit via de file manager driver
A2EA-	AD 75 AA	LDA	\$AA75	CLOSE commando afhandeling *****
A2ED-	C9 A0	CMP	#\$A0	Is geen filenaam opgegeven,
A2EF-	F0 25	BEQ	\$A316	sluit dan alle files af.
A2F1-	20 64 A7	JSR	\$A764	Of zoek open file af naar filenaam
A2F4-	B0 3A	BCS	\$A330	en is file niet open, dan exit
A2F6-	20 FC A2	JSR	\$A2FC	close file en maak buffer vrij
A2F9-	4C EA A2	JMP	\$A2EA	Nu terug voor na-controle buffers
A2FC-	20 AF A7	JSR	\$A7AF	Is dit een EXEC buffer ?
A2FF-	D0 05	BNE	\$A306	Als dat zo is, dan EXEC flag afzetten
A301-	A9 00	LDA	#\$00	Dag flag!

A303-	8D B3 AA	STA	\$AAB3	
A306-	A0 00	LDY	##00	Maak nu de buffer vrij
A308-	98	TYA		door een 0 te plaatsen
A309-	91 40	STA	(\$40),Y	in het filenaam field
A30B-	20 4E A7	JSR	\$A74E	Copiëer de stand van zaken in
A30E-	A9 02	LDA	##02	de bekende parmlist en set
A310-	8D BB B5	STA	\$B5BB	de file manager opcode op CLOSE.
A313-	4C AB A6	JMP	\$A6AB	Exit via file manager driver
A316-	20 92 A7	JSR	\$A792	CLOSE ALL FILES *****
A319-	D0 05	BNE	\$A320	Wijs naar eerste buffer; is het EXEC?
A31B-	20 9A A7	JSR	\$A79A	Wijs naar volgende filebuffer
A31E-	F0 10	BEQ	\$A330	Geen filenaambuffers meer, dan exit!
A320-	20 AF A7	JSR	\$A7AF	EXEC's ?
A323-	F0 F6	BEQ	\$A31B	Ja!
A325-	20 AA A7	JSR	\$A7AA	Wordt hij niet gebruikt ?
A32B-	F0 F1	BEQ	\$A31B	Toch wel!
A32A-	20 FC A2	JSR	\$A2FC	Want anders CLOSE en schoonmaken.
A32D-	4C 16 A3	JMP	\$A316	Nu nog een keer overdoen.
A330-	60	RTS		+die kun je maar nodig hebben!
A331-	A9 09	LDA	##09	BSAVE afhandeling *****
A333-	2D 65 AA	AND	\$AA65	Eerst moet worden uitgezocht of de
A336-	C9 09	CMP	##09	A en L parameter aanwezig is.
A338-	F0 03	BEQ	\$A33D	Anders zal een foutmelding
A33A-	4C 00 A0	JMP	\$A000	volgen: "SYNTAX ERROR"
A33D-	A9 04	LDA	##04	Open file en
A33F-	20 D5 A3	JSR	\$A3D5	bekijk of het een Binary file is.
A342-	AD 73 AA	LDA	\$AA73	Nu alle adres parameters opzetten.
A345-	AC 72 AA	LDY	\$AA72	
A34B-	20 E0 A3	JSR	\$A3E0	Eerst de A afwerken
A34B-	AD 6D AA	LDA	\$AA6D	
A34E-	AC 6C AA	LDY	\$AA6C	
A351-	20 E0 A3	JSR	\$A3E0	Vervolgens de L
A354-	AD 73 AA	LDA	\$AA73	
A357-	AC 72 AA	LDY	\$AA72	
A35A-	4C FF A3	JMP	\$A3FF	Nu wordt de file getransporteerd.
A35D-	20 AB A2	JSR	\$A2AB	BLOAD afhandeling *****
A360-	A9 7F	LDA	##7F	Is het een Binary file?
A362-	2D C2 B5	AND	\$B5C2	Eerst controleren
A365-	C9 04	CMP	##04	anders volgt een foutmelding.
A367-	F0 03	BEQ	\$A36C	
A369-	4C D0 A6	JMP	\$A6D0	"FILE TYPE MISMATCH"
A36C-	A9 04	LDA	##04	Open de "B" file
A36E-	20 D5 A3	JSR	\$A3D5	en test het file type.
A371-	20 7A A4	JSR	\$A47A	Nu wordt de adreswaarde
A374-	AA	TAX		en de lengte van de file
A375-	AD 65 AA	LDA	\$AA65	afgehandeld.
A37B-	29 01	AND	##01	
A37A-	D0 06	BNE	\$A3B2	
A37C-	8E 72 AA	STX	\$AA72	Adres
A37F-	8C 73 AA	STY	\$AA73	Adres +1
A382-	20 7A A4	JSR	\$A47A	
A385-	AE 72 AA	LDX	\$AA72	
A38B-	AC 73 AA	LDY	\$AA73	
A38B-	4C 71 A4	JMP	\$A471	Lees de hoeveelheid bytes af.
A38E-	20 5D A3	JSR	\$A35D	BRUN commando afhandeling *****
A391-	20 51 AB	JSR	\$AB51	Eerst Bload en dan pointers opzetten.
A394-	6C 72 AA	JMP	(\$AA72)	Exit met jump naar A waarde programma
A397-	AD B6 AA	LDA	\$AAB6	SAVE commando afhandeling *****
A39A-	F0 20	BEQ	\$A3BC	Stel Basic type vast. Is het Integer?
A39C-	A5 D6	LDA	\$D6	+Is het programma beveiligd?!
A39E-	10 03	BPL	\$A3A3	Anders volgt "fout"-melding:
A3A0-	4C CC A6	JMP	\$A6CC	"PROGRAM TOO LARGE"

A3A3-	A9 02	LDA	##02	Is alles echter OK, dan
A3A5-	20 D5 A3	JSR	\$A3D5	openen we de file en testen we "A" type.
A3A8-	38	SEC		
A3A9-	A5 AF	LDA	\$AF	Bereken nu het einde van het programma
A3AB-	E5 67	SBC	\$67	
A3AD-	A8	TAY		
A3AE-	A5 B0	LDA	\$B0	
A3B0-	E5 68	SBC	\$68	Nu weten we de omvang!
A3B2-	20 E0 A3	JSR	\$A3E0	En houdt de administratie ervan bij.
A3B5-	A5 68	LDA	\$68	
A3B7-	A4 67	LDY	\$67	
A3B9-	4C FF A3	JMP	\$A3FF	Exit door LOMEM vast te leggen.
A3BC-	A9 01	LDA	##01	Open en test file type, indien
A3BE-	20 D5 A3	JSR	\$A3D5	Integer Basic wordt verwacht.
A3C1-	38	SEC		
A3C2-	A5 4C	LDA	\$4C	Bereken HIMEM
A3C4-	E5 CA	SBC	\$CA	en de start van het programma
A3C6-	A8	TAY		
A3C7-	A5 4D	LDA	\$4D	
A3C9-	E5 CB	SBC	\$CB	Zodat de omvang daaruit volgt.
A3CB-	20 E0 A3	JSR	\$A3E0	Schrijf 2 bytes naar de file
A3CE-	A5 CB	LDA	\$CB	
A3D0-	A4 CA	LDY	\$CA	
A3D2-	4C FF A3	JMP	\$A3FF	Nu wordt het programma weggeschreven.
A3D5-	8D C2 B5	STA	\$B5C2	OPEN EN TEST Subroutine *****
A3D8-	48	PHA		Zet gewenst filetype in de parmlist
A3D9-	20 AB A2	JSR	\$A2AB	Open de file
A3DC-	68	PLA		
A3DD-	4C C4 A7	JMP	\$A7C4	Controleer het filetype
A3E0-	8C C1 B5	STY	\$B5C1	Schrijf 2 bytes waarde naar file
A3E3-	8C C3 B5	STY	\$B5C3	Data omvang in f.m. parmlist
A3E6-	8D C2 B5	STA	\$B5C2	
A3E9-	A9 04	LDA	##04	Set write opcode
A3EB-	8D BB B5	STA	\$B5BB	voor een byte
A3EE-	A9 01	LDA	##01	
A3F0-	8D BC B5	STA	\$B5BC	
A3F3-	20 AB A6	JSR	\$A6AB	file manager driver werkt byte af
A3F6-	AD C2 B5	LDA	\$B5C2	Nu herhaling
A3F9-	8D C3 B5	STA	\$B5C3	voor tweede byte
A3FC-	4C AB A6	JMP	\$A6AB	file manager driver werkt byte af.
A3FF-	8C C3 B5	STY	\$B5C3	Read/Write range van bytes. *****
A402-	8D C4 B5	STA	\$B5C4	Set de rangeadressen in f.m. parmlist
A405-	A9 02	LDA	##02	
A407-	4C B6 B6	JMP	\$B6B6	Verify patch - data goed geschreven?
A40A-	20 AB A6	JSR	\$A6AB	file manager driver
A40D-	4C EA A2	JMP	\$A2EA	Close de handling
A410-	4C D0 A6	JMP	\$A6D0	"FILE TYPE MISMATCH"!
A413-	20 16 A3	JSR	\$A316	LOAD commando afhandeling *****
A416-	20 AB A2	JSR	\$A2AB	Open de betreffende file
A419-	A9 23	LDA	##23	Is het een "A" of "I" type ? ROM/RAM?
A41B-	2D C2 B5	AND	\$B5C2	
A41E-	F0 F0	BEQ	\$A410	Hupsakee, foutmelding!
A420-	8D C2 B5	STA	\$B5C2	Filetype ligt vast
A423-	AD B6 AA	LDA	\$AAB6	Maar welk Basic dialect is actief?
A426-	F0 28	BEQ	\$A450	Als het Integer is, dan A450
A428-	A9 02	LDA	##02	
A42A-	20 B1 A4	JSR	\$A4B1	Het wordt Applesoft, hopelijk aanwezig!
A42D-	20 7A A4	JSR	\$A47A	Lengte van het programma
A430-	18	CLC		Nu gaan we alle pointers verzorgen
A431-	65 67	ADC	\$67	Vergelijking LOMEM met
A433-	AA	TAX		HIMEM, om te zien of programma niet
A434-	98	TYA		te lang is.

A435-	65 68	ADC	\$68	
A437-	C5 74	CMP	\$74	
A439-	B0 70	BCS	\$A4AB	Zie je wel, "PROGRAM TOO LARGE"//
A43B-	85 B0	STA	\$B0	
A43D-	85 6A	STA	\$6A	Set programma pointers
A43F-	86 AF	STX	\$AF	
A441-	86 69	STX	\$69	
A443-	A6 67	LDX	\$67	
A445-	A4 68	LDY	\$68	
A447-	20 71 A4	JSR	\$A471	Read range van bytes
A44A-	20 51 A8	JSR	\$A851	reset de DOS pointers
A44D-	6C 60 9D	JMP	(\$9D60)	en ga naar conversie routine
A450-	A9 01	LDA	##01	LOAD INTEGER programma *****
A452-	20 B1 A4	JSR	\$A4B1	kies het Integer dialekt
A455-	20 7A A4	JSR	\$A47A	Lees de eerste 2 bytes van programma
A458-	38	SEC		
A459-	A5 4C	LDA	\$4C	Bepaal de omvang
A45B-	ED 60 AA	SBC	\$AA60	door HIMEM-Proglengte
A45E-	AA	TAX		
A45F-	A5 4D	LDA	\$4D	
A461-	ED 61 AA	SBC	\$AA61	Past het allemaal ?
A464-	90 45	BCC	\$A4AB	"PROGRAM TOO LARGE"
A466-	AB	TAY		
A467-	C4 4B	CPY	\$4B	Bekijk LOMEM
A469-	90 40	BCC	\$A4AB	"PROGRAM TOO LARGE"
A46B-	F0 3E	BEQ	\$A4AB	Minder dan LOMEM, dan ook goed fout!
A46D-	84 CB	STY	\$CB	Indien alles goed, dan setten
A46F-	86 CA	STX	\$CA	we de programmastart.
A471-	8E C3 B5	STX	\$B5C3	Lees/Schrijf bepaalde omvang bytes
A474-	8C C4 B5	STY	\$B5C4	
A477-	4C 0A A4	JMP	\$A40A	naar file manager driver
A47A-	AD 0A 9D	LDA	\$9D0A	Lees 2 bytes uit een file *****
A47D-	8D C3 B5	STA	\$B5C3	
A480-	AD 0B 9D	LDA	\$9D0B	
A483-	8D C4 B5	STA	\$B5C4	parmlist gereed maken
A486-	A9 00	LDA	##00	
A488-	8D C2 B5	STA	\$B5C2	
A48B-	A9 02	LDA	##02	
A48D-	8D C1 B5	STA	\$B5C1	
A490-	A9 03	LDA	##03	
A492-	8D BB B5	STA	\$B5BB	
A495-	A9 02	LDA	##02	
A497-	8D BC B5	STA	\$B5BC	
A49A-	20 AB A6	JSR	\$A6AB	Call file manager driver.
A49D-	AD 61 AA	LDA	\$AA61	
A4A0-	8D C2 B5	STA	\$B5C2	Zet de omvang, die werd berekend
A4A3-	AB	TAY		in parmlist.
A4A4-	AD 60 AA	LDA	\$AA60	
A4A7-	8D C1 B5	STA	\$B5C1	
A4AA-	60	RTS		----
A4AB-	20 EA A2	JSR	\$A2EA	CLOSE file
A4AE-	4C CC A6	JMP	\$A6CC	en print "PROGRAM TOO LARGE"
A4B1-	CD C2 B5	CMP	\$B5C2	BASIC keuze subroutine *****
A4B4-	F0 1A	BEQ	\$A4D0	Klopt het BASIC dialekt, dan klaar
A4B6-	AE 5F AA	LDX	\$AA5F	zet de commando index weg voor het
A4B9-	8E 62 AA	STX	\$AA62	geval APPLESOFT gerund moet worden.
A4BC-	4A	LSR		
A4BD-	F0 03	BEQ	\$A4C2	
A4BF-	4C 9E A5	JMP	\$A59E	We kiezen Integer Basic
A4C2-	A2 1D	LDX	##1D	Nu eerst filenaam veilig stellen,
A4C4-	BD 75 AA	LDA	\$AA75,X	voor het geval er eerst het programma
A4C7-	9D 93 AA	STA	\$AA93,X	"APPLESOFT" gerund moet worden voor

A4CA-	CA	DEX		status "Applesoft in RAM"
A4CB-	10 F7	BPL	\$A4C4	Doe "Applesoft" als filenaam
A4CD-	4C 7A A5	JMP	\$A57A	is weggeschreven.
A4D0-	60	RTS		-----
A4D1-	AD B6 AA	LDA	\$AAB6	RUN afhandeling *****
A4D4-	F0 03	BEQ	\$A4D9	Kies Basic en zet RUN intercept flag
A4D6-	BD B7 AA	STA	\$AAB7	zo, dat RUN afgehandeld wordt.
A4D9-	20 13 A4	JSR	\$A413	Ga naar Load
A4DC-	20 C8 9F	JSR	\$9FCB	Verspring een regel op het scherm
A4DF-	20 51 A8	JSR	\$A851	Reset de diverse pointers
A4E2+	6C 58 9D	JMP	(\$9D5B)	RUN startpunt voor vigerend Basic +
A4E5-	A5 4A	LDA	\$4A	Integer Basic RUN afhandeling
A4E7-	B5 CC	STA	\$CC	
A4E9-	A5 4B	LDA	\$4B	Voer een CLR manoeuvre uit.
A4EB-	B5 CD	STA	\$CD	
A4ED-	6C 56 9D	JMP	(\$9D56)	Integer CHAIN beginpunt
A4F0-	20 16 A4	JSR	\$A416	CHAIN commando afhandeling *****
A4F3-	20 C8 9F	JSR	\$9FCB	regel verspringen na Load
A4F6-	20 51 A8	JSR	\$A851	DOS intercept pointers setten
A4F9-	6C 56 9D	JMP	(\$9D56)	ga naar het beginpunt.
A4FC-	20 65 D6	JSR	\$D665	APPLESOFT ROM RUN entry point ***
A4FF-	B5 33	STA	\$33	A.S. Chain is maar een Run!
A501-	B5 D8	STA	\$D8	reset Onerr
A503-	4C D2 D7	JMP	\$D7D2	clear variabelen en naar entry point
A506-	20 65 0E	JSR	\$0E65	APPLESOFT RAM RUN
A509-	B5 33	STA	\$33	clear variabelen
A50B-	B5 D8	STA	\$D8	reset Onerr
A50D-	4C D4 0F	JMP	\$0FD4	naar Run entry point
A510-	20 26 A5	JSR	\$A526	WRITE commando afhandeling *****
A513-	A9 05	LDA	#\$05	set write mode
A515-	BD 52 AA	STA	\$AA52	
A518-	4C B3 9F	JMP	\$9FB3	Exit DOS.
A51B-	20 26 A5	JSR	\$A526	READ commando afhandeling *****
A51E-	A9 01	LDA	#\$01	set read mode
A520-	BD 51 AA	STA	\$AA51	
A523-	4C B3 9F	JMP	\$9FB3	Exit DOS.
A526-	20 64 A7	JSR	\$A764	Gemeenschappelijke READ/WRITE code**
A529-	90 06	BCC	\$A531	Open file buffer indien niet open!
A52B-	20 A3 A2	JSR	\$A2A3	
A52E-	4C 34 A5	JMP	\$A534	
A531-	20 4E A7	JSR	\$A74E	Copiëer file buffer werkruimte naar
A534-	AD 65 AA	LDA	\$AA65	f.m.; controleer "R" of "B" type
A537-	29 06	AND	#\$06	indien niet,
A539-	F0 13	BEQ	\$A54E	exit via Rts
A53B-	A2 03	LDX	#\$03	Anders moeten de operands
A53D-	BD 6E AA	LDA	\$AA6E,X	worden weggeschreven in de parmlist
A540-	9D BD B5	STA	\$B5BD,X	
A543-	CA	DEX		
A544-	10 F7	BPL	\$A53D	
A546-	A9 0A	LDA	#\$0A	Omdat kennelijk "R" of "B" vigeert,
A54B-	BD BB B5	STA	\$B5BB	moet een Position commando worden
A54B-	20 A8 A6	JSR	\$A6A8	afgehandeld voor read of write!
A54E-	60	RTS		----
A54F-	A9 40	LDA	#\$40	INIT commando afhandeling *****
A551-	2D 65 AA	AND	\$AA65	Bekijk de "V" parameter.
A554-	F0 05	BEQ	\$A55B	
A556-	AD 66 AA	LDA	\$AA66	"V" ? {>0}
A559-	D0 05	BNE	\$A560	Anders moeten wij een V
A55B-	A9 FE	LDA	#\$FE	toekennen - 254.
A55D-	BD 66 AA	STA	\$AA66	
A560-	AD 0D 9D	LDA	\$9D0D	Haal het DOS Pagegetal op -\$9D/48K
A563-	BD BC B5	STA	\$B5BC	en berg hem op in het subcodelijstje.

+ Aangrijppunt voor 'beveiligingen'!

A566-	A9 0B	LDA	#\$0B	
A568-	20 AA A2	JSR	\$A2AA	Call file manager voor INIT aktie.
A56B-	4C 97 A3	JMP	\$A397	Save het begroetingsprogramma ("Hello")
A56E-	A9 06	LDA	#\$06	CATALOG commando afhandeling *****
A570-	20 AA A2	JSR	\$A2AA	Verzorg de Catalog opcode
A573-	AD BF B5	LDA	\$B5BF	Haal het Volumenummer op
A576-	8D 66 AA	STA	\$AA66	en set nieuwe Volnummer voor verder gebruik.
A579-	60	RTS		
A57A-	A9 4C	LDA	#\$4C	FP commando afhandeling *****
A57C-	20 B2 A5	JSR	\$A5B2	Set Basic type voor ROM Card
A57F-	F0 2E	BEQ	\$A5AF	Gelukt, dan naar Coldstart
A581-	A9 00	LDA	#\$00	Set anders Integer Basic
A583-	8D B6 AA	STA	\$AAB6	
A586-	A0 1E	LDY	#\$1E	en probeer Applesoft in RAM te laden
A588-	20 97 A0	JSR	\$A097	("Applesoft" is een Integer programma
A58B-	A2 09	LDX	#\$09	op de masterdiskette)
A58D-	BD B7 AA	LDA	\$AAB7,X	
A590-	9D 74 AA	STA	\$AA74,X	
A593-	CA	DEX		(Als het programma niet op de disk
A594-	D0 F7	BNE	\$A58D	voorkomt, dan volgt foutmelding)
A596-	A9 C0	LDA	#\$C0	
A598-	8D 51 AA	STA	\$AA51	
A59B-	4C D1 A4	JMP	\$A4D1	
A59E-	A9 20	LDA	#\$20	INT commando afhandeling *****
A5A0-	20 B2 A5	JSR	\$A5B2	Set Basic type
A5A3-	F0 05	BEQ	\$A5AA	
A5A5-	A9 01	LDA	#\$01	
A5A7-	4C D2 A6	JMP	\$A6D2	"LANGUAGE NOT AVAILABLE"
A5AA-	A9 00	LDA	#\$00	
A5AC-	8D B7 AA	STA	\$AAB7	
A5AF-	4C 84 9D	JMP	\$9DB4	Naar Coldstart
A5B2-	CD 00 E0	CMP	\$E000	Set ROM voor gewenst Basic dialect
A5B5-	F0 0E	BEQ	\$A5C5	Integer heeft \$20 in A
A5B7-	8D 80 C0	STA	\$C080	Applesoft \$4C in A
A5BA-	CD 00 E0	CMP	\$E000	Deze komen voor in \$E000
A5BD-	F0 06	BEQ	\$A5C5	Is juiste Basic geactiveerd ?
A5BF-	8D 81 C0	STA	\$C081	Probeer opnieuw de ROM's
A5C2-	CD 00 E0	CMP	\$E000	Nog niet juist, dan foutcode.
A5C5-	60	RTS		----
A5C6-	20 A3 A2	JSR	\$A2A3	EXEC commando afhandeling *****
A5C9-	AD 4F AA	LDA	\$AA4F	
A5CC-	8D B4 AA	STA	\$AAB4	Copiëer file buffer adres
A5CF-	AD 50 AA	LDA	\$AA50	
A5D2-	8D B5 AA	STA	\$AAB5	
A5D5-	AD 75 AA	LDA	\$AA75	
A5D8-	8D B3 AA	STA	\$AAB3	Set EXEC flag
A5DB-	D0 0E	BNE	\$A5EB	Let op de "R" parameter
A5DD-	20 64 A7	JSR	\$A764	POSITION afhandeling *****
A5E0-	90 06	BCC	\$A5EB	Lokaliseer open file buffer
A5E2-	20 A3 A2	JSR	\$A2A3	of open er een als textfile
A5E5-	4C EB A5	JMP	\$A5EB	
A5E8-	20 4E A7	JSR	\$A74E	Copiëer de pointers in parmlist
A5EB-	AD 65 AA	LDA	\$AA65	Haal opcode op - is het "R"
A5EE-	29 04	AND	#\$04	Niet gegeven ?
A5F0-	F0 1B	BEQ	\$A60D	Goed zo!
A5F2-	AD 6E AA	LDA	\$AA6E	Zoek naar de "R"-de record.
A5F5-	D0 08	BNE	\$A5FF	
A5F7-	AE 6F AA	LDX	\$AA6F	
A5FA-	F0 11	BEQ	\$A60D	Was-ie 0!
A5FC-	CE 6F AA	DEC	\$AA6F	Verminder "R" met 1
A5FF-	CE 6E AA	DEC	\$AA6E	
A602-	20 BC A6	JSR	\$A68C	Lees file byte voor byte

A605-	F0 38	BEQ	\$A63F	Is het eind bereikt, dan foutmelding!
A607-	C9 8D	CMP	##8D	Haha, een return.
A609-	D0 F7	BNE	\$A602	Nog een keer. Komt er een \$8D?
A60B-	F0 E5	BEQ	\$A5F2	Skip volgende record.
A60D-	60	RTS		----
A60E-	20 5E A6	JSR	\$A65E	WRITE 1 Data byte naar de file
A611-	B0 66	BCS	\$A679	Basic moet werken. Anders Warmstart.
A613-	AD 5C AA	LDA	\$AA5C	Handel de parmlist af, zodat er
A616-	8D C3 B5	STA	\$B5C3	een byte kan worden weggeschreven.
A619-	A9 04	LDA	##04	
A61B-	8D BB B5	STA	\$B5BB	
A61E-	A9 01	LDA	##01	
A620-	8D BC B5	STA	\$B5BC	
A623-	4C AB A6	JMP	\$A6A8	Call file manager en stop
A626-	20 5E A6	JSR	\$A65E	READ 1 byte uit een file
A629-	B0 4E	BCS	\$A679	Basic moet werken. Anders Warmstart.
A62B-	A9 06	LDA	##06	CSW code 6 (laat prompt weg)
A62D-	8D 52 AA	STA	\$AA52	
A630-	20 8C A6	JSR	\$A68C	lees file verder door.
A633-	D0 0F	BNE	\$A644	Test of dit een kleine letter is ?
A635-	20 FC A2	JSR	\$A2FC	EOF - dus Close file
A638-	A9 03	LDA	##03	Is de statuscode 3 (EXEC)!!?
A63A-	CD 52 AA	CMP	\$AA52	Anders komen er problemen...
A63D-	F0 CE	BEQ	\$A60D	
A63F-	A9 05	LDA	##05	
A641-	4C D2 A6	JMP	\$A6D2	"END OF DATA"
A644-	C9 E0	CMP	##E0	Is het ingelezen karakter lowercase?
A646-	90 02	BCC	\$A64A	Dan moet de GETIN monitor routine
A648-	29 7F	AND	##7F	worden misleid.
A64A-	8D 5C AA	STA	\$AA5C	Zet het meest significante bit af.
A64D-	AE 5A AA	LDX	\$AA5A	Zet het databyte weg.
A650-	F0 09	BEQ	\$A65B	Exit DOS
A652-	CA	DEX		Gebruik de index en zet hi-bit weer
A653-	BD 00 02	LDA	\$0200,X	aan van de data-byte in de line-buffer.
A656-	09 80	ORA	##80	Zo kan het weer als onderkast karakter
A658-	9D 00 02	STA	\$0200,X	worden aangemerkt bij de output.
A65B-	4C B3 9F	JMP	\$9FB3	Exit DOS.
A65E-	48	PHA		Is Basic actief?
A65F-	AD B6 AA	LDA	\$AAB6	Welk Basic ?
A662-	F0 0E	BEQ	\$A672	Integer ? Ja!
A664-	A6 76	LDX	\$76	Applesoft - maar op welke manier?
A666-	E8	INX		Is AS in de immediate mode ?
A667-	F0 0D	BEQ	\$A676	
A669-	A6 33	LDX	\$33	Dan moet er een prompt zijn.
A66B-	E0 DD	CPX	##DD	
A66D-	F0 07	BEQ	\$A676	
A66F-	68	PLA		Anders executeert Basic een
A670-	18	CLC		programma.
A671-	60	RTS		-----
A672-	A5 D9	LDA	\$D9	Is de computer met een programma bezig.
A674-	30 F9	BMI	\$A66F	Is de INT Run mode flag geset ?
A676-	68	PLA		
A677-	38	SEC		Keer dan terug met de goed code.
A678-	60	RTS		-----
A679-	20 FC A2	JSR	\$A2FC	Close onderhavige file
A67C-	20 5B A7	JSR	\$A75B	
A67F-	4C B3 9F	JMP	\$9FB3	DOS exit
A682-	20 9D A6	JSR	\$A69D	EXEC leest 1 byte uit file.
A685-	20 4E A7	JSR	\$A74E	werk parmlist bij.
A688-	A9 03	LDA	##03	
A68A-	D0 A1	BNE	\$A62D	
A68C-	A9 03	LDA	##03	Lees een karakter uit een textfile

A68E-	8D BB B5	STA	\$B5BB	Werk parameter lijst bij voor
A691-	A9 01	LDA	##01	het lezen van een byte.
A693-	8D BC B5	STA	\$B5BC	
A696-	20 A8 A6	JSR	\$A6A8	Call f.m. driver
A699-	AD C3 B5	LDA	\$B5C3	Haal de data omvang op
A69C-	60	RTS		----
A69D-	AD B5 AA	LDA	\$AAB5	Set pointers, die verwijzen
A6A0-	85 41	STA	\$41	naar de EXEC buffer
A6A2-	AD B4 AA	LDA	\$AAB4	
A6A5-	85 40	STA	\$40	
A6A7-	60	RTS		-----
A6A8-	20 06 AB	JSR	\$AB06	File Manager Driver subroutine *****
A6AB-	90 16	BCC	\$A6C3	Call file manager zelf en controleer
A6AD-	AD C5 B5	LDA	\$B5C5	op fouten. Haal returncode op.
A6B0-	C9 05	CMP	##05	
A6B2-	F0 03	BEQ	\$A6B7	
A6B4-	4C 5E B6	JMP	\$B65E	naar Append patch
A6B7-	4C 92 B6	JMP	\$B692	
A6BA-	EA	NOP		Was de fout niet "End of Data"
A6BB-	EA	NOP		dan doen we net, of er Ø was
A6BC-	EA	NOP		gelezen.
A6BD-	EA	NOP		
A6BE-	A2 00	LDX	##00	Zet Ø in file name field
A6C0-	8E C3 B5	STX	\$B5C3	Terug naar aanroepende routine
A6C3-	60	RTS		-----
A6C4-	A9 0B	LDA	##0B	Code voor "Syntax Error"
A6C6-	D0 0A	BNE	\$A6D2	Voer melding uit.
A6C8-	A9 0C	LDA	##0C	Code voor "No Buffers Available"
A6CA-	D0 06	BNE	\$A6D2	Voer uit.
A6CC-	A9 0E	LDA	##0E	Code voor "Program Too Large"
A6CE-	D0 02	BNE	\$A6D2	Voer uit.
A6D0-	A9 0D	LDA	##0D	Code voor "File Type Mismatch"
A6D2-	8D 5C AA	STA	\$AA5C	Foutafhandelings werkrountine *****
A6D5-	20 E6 BF	JSR	\$BFE6	Set Warmstart flag en clear status
A6D8-	AD B6 AA	LDA	\$AAB6	Welk Basic actief ?
A6DB-	F0 04	BEQ	\$A6E1	Moet foutmelding nu volgen ?
A6DD-	A5 D8	LDA	\$D8	Is Applesoft Onerr actief, dan
A6DF-	30 0E	BMI	\$A6EF	naar \$A6EF
A6E1-	A2 00	LDX	##00	Anders volgt nu een fout-
A6E3-	20 02 A7	JSR	\$A702	melding: \$A702
A6E6-	AE 5C AA	LDX	\$AA5C	{return bel return}
A6E9-	20 02 A7	JSR	\$A702	Bestemd voor Integer en AS zonder
A6EC-	20 C8 9F	JSR	\$9FC8	Onerr conditie.
A6EF-	20 51 AB	JSR	\$AB51	Herstel DOS Intercept pointers
A6F2-	20 5E A6	JSR	\$A65E	Is er een Basic programma
A6F5-	AE 5C AA	LDX	\$AA5C	onderbroken, geef dan de code
A6F8-	A9 03	LDA	##03	door aan de Basic fout
A6FA-	B0 03	BCS	\$A6FF	afhandelingsroutine.
A6FC-	6C 5A 9D	JMP	(\$9D5A)	
A6FF-	6C 5E 9D	JMP	(\$9D5E)	Of doe een Warmstart.
A702-	BD 3F AA	LDA	\$AA3F,X	Haal de foutmeldingscode offset op.
A705-	AA	TAX		
A706-	8E 63 AA	STX	\$AA63	Daar komt de foutmelding!
A709-	BD 71 A9	LDA	\$A971,X	Hij "rolt" naar binnen
A70C-	48	PHA		
A70D-	09 80	ORA	##80	en het laatste teken heeft hi-bit aan.
A70F-	20 C5 9F	JSR	\$9FC5	Nu kun je hem zien: COUNT
A712-	AE 63 AA	LDX	\$AA63	Even de index (teller) bijwerken
A715-	E8	INX		+1
A716-	68	PLA		
A717-	10 ED	BPL	\$A706	en loop terug; volgend teken.
A719-	60	RTS		Klaar ----

A71A-	AD 66 AA	LDA	\$AA66	Copiëer parameters naar f.m. parmlist ***
A71D-	8D BF B5	STA	\$B5BF	volume
A720-	AD 68 AA	LDA	\$AA68	
A723-	8D C0 B5	STA	\$B5C0	drive
A726-	AD 6A AA	LDA	\$AA6A	
A729-	8D C1 B5	STA	\$B5C1	slot
A72C-	AD 06 9D	LDA	\$9D06	
A72F-	8D C3 B5	STA	\$B5C3	adres primaire filenaam buffer
A732-	AD 07 9D	LDA	\$9D07	
A735-	8D C4 B5	STA	\$B5C4	adres primaire filenaam buffer+1
A738-	A5 40	LDA	\$40	
A73A-	8D 4F AA	STA	\$AA4F	adres filebuffer
A73D-	A5 41	LDA	\$41	
A73F-	8D 50 AA	STA	\$AA50	adres filebuffer+1
A742-	60	RTS		-----
A743-	A0 1D	LDY	#\$1D	Copiëer filenaam naar buffer ***
A745-	B9 75 AA	LDA	\$AA75,Y	
A748-	91 40	STA	(\$40),Y	copiëer naam in juiste field
A74A-	88	DEY		
A74B-	10 FB	BPL	\$A745	
A74D-	60	RTS		-----
A74E-	A0 1E	LDY	#\$1E	copiëer buffer pointers naar parmlist
A750-	B1 40	LDA	(\$40),Y	uit file manager werkruimte.
A752-	99 A9 B5	STA	\$B5A9,Y	Track/sector List buffer pointer
A755-	C8	INY		Data sector buffer adres
A756-	C0 26	CPY	#\$26	Volgende filebuffer linkadres
A758-	D0 F6	BNE	\$A750	
A75A-	60	RTS		-----
A75B-	A0 00	LDY	#\$00	Reset Status 0 en set Warmstart ***
A75D-	8C 51 AA	STY	\$AA51	
A760-	8C 52 AA	STY	\$AA52	
A763-	60	RTS		-----
A764-	A9 00	LDA	#\$00	Lokaliseer geschikte file buffer ***
A766-	85 45	STA	\$45	Veronderstel 0 buffers vrij.
A768-	20 92 A7	JSR	\$A792	Controleer 1ste file buffer
A76B-	4C 73 A7	JMP	\$A773	en kijk of hij beschikbaar is.
A76E-	20 9A A7	JSR	\$A79A	Laat \$40-41 verwijzen naar volgende
A771-	F0 1D	BEQ	\$A790	buffer in de reeks. Geen in gebruik?
A773-	20 AA A7	JSR	\$A7AA	Is er een buffer, controleer dan
A776-	D0 0A	BNE	\$A7B2	1ste byte file naam field.
A77B-	A5 40	LDA	\$40	Indien 0 zet file naam adres weg
A77A-	85 44	STA	\$44	in \$44
A77C-	A5 41	LDA	\$41	
A77E-	85 45	STA	\$45	en \$45.
A780-	D0 EC	BNE	\$A76E	Volgende buffer - branch altijd!
A782-	A0 1D	LDY	#\$1D	Vergelijk de filenamen
A784-	B1 40	LDA	(\$40),Y	
A786-	D9 75 AA	CMP	\$AA75,Y	
A789-	D0 E3	BNE	\$A76E	Niet gelijk, dan volgende buffer.
A78B-	88	DEY		
A78C-	10 F6	BPL	\$A7B4	Loop terug voor verdere controle.
A78E-	18	CLC		Return met file open code.
A78F-	60	RTS		----
A790-	38	SEC		Niet in gebruik.
A791-	60	RTS		----
A792-	AD 00 9D	LDA	\$9D00	\$40-41 gaat verwijzen naar
A795-	AE 01 9D	LDX	\$9D01	eerste file buffer in reeks.
A798-	D0 0A	BNE	\$A7A4	
A79A-	A0 25	LDY	#\$25	\$40-41 gaat verwijzen naar
A79C-	B1 40	LDA	(\$40),Y	volgende file buffer in reeks.
A79E-	F0 09	BEQ	\$A7A9	
A7A0-	AA	TAX		

A7A1-	88	DEY		
A7A2-	B1 40	LDA	(\$40),Y	
A7A4-	86 41	STX	\$41	Adressen verzorgd.
A7A6-	85 40	STA	\$40	
A7A8-	8A	TXA		
A7A9-	60	RTS		-----
A7AA-	A0 00	LDY	#\$00	Haal eerste byte uit
A7AC-	B1 40	LDA	(\$40),Y	file naam buffer op en
A7AE-	60	RTS		zet hem in de buffer.
A7AF-	AD B3 AA	LDA	\$AAB3	Wordt die buffer gebruikt
A7B2-	F0 0E	BEQ	\$A7C2	voor een EXEC status ? Nee ?
A7B4-	AD B4 AA	LDA	\$AAB4	Vergelijk dan de EXEC buffer adressen
A7B7-	C5 40	CMP	\$40	
A7B9-	D0 08	BNE	\$A7C3	
A7BB-	AD B5 AA	LDA	\$AAB5	
A7BE-	C5 41	CMP	\$41	met de file buffer
A7C0-	F0 01	BEQ	\$A7C3	en keer terug met juiste code.
A7C2-	CA	DEX		
A7C3-	60	RTS		-----
A7C4-	4D C2 B5	EOR	\$B5C2	Controleer het filetype.
A7C7-	F0 0A	BEQ	\$A7D3	OK, dan klaar.
A7C9-	29 7F	AND	#\$7F	Zet lock-bit af en test opnieuw.
A7CB-	F0 06	BEQ	\$A7D3	OK, dan klaar.
A7CD-	20 EA A2	JSR	\$A2EA	Close file en geef
A7D0-	4C D0 A6	JMP	\$A6D0	"FILE TYPE MISMATCH"
A7D3-	60	RTS		-----
A7D4-	38	SEC		Initialiseer file buffer keten.
A7D5-	AD 00 9D	LDA	\$9D00	Laat \$40-41 naar 1ste buffer
A7D8-	85 40	STA	\$40	verwijzen.
A7DA-	AD 01 9D	LDA	\$9D01	
A7DD-	85 41	STA	\$41	
A7DF-	AD 57 AA	LDA	\$AA57	Zet indexteller op
A7E2-	8D 63 AA	STA	\$AA63	MAXFILES waarde.
A7E5-	A0 00	LDY	#\$00	Zet een 0 in het file naam field
A7E7-	98	TYA		opdat deze wordt vrijgemaakt.
A7E8-	91 40	STA	(\$40),Y	
A7EA-	A0 1E	LDY	#\$1E	Zet de linkpointers op in buffer,
A7EC-	38	SEC		zodat ze naar de file verwijzen.
A7ED-	A5 40	LDA	\$40	
A7EF-	E9 2D	SBC	#\$2D	
A7F1-	91 40	STA	(\$40),Y	
A7F3-	48	PHA		
A7F4-	A5 41	LDA	\$41	
A7F6-	E9 00	SBC	#\$00	
A7F8-	C8	INY		
A7F9-	91 40	STA	(\$40),Y	
A7FB-	AA	TAX		Zet pointer op t.b.v. de
A7FC-	CA	DEX		Track en Sector List buffer
A7FD-	68	PLA		-256 bytes van file manager
A7FE-	48	PHA		werkruimte.
A7FF-	C8	INY		
A800-	91 40	STA	(\$40),Y	
A802-	8A	TXA		
A803-	C8	INY		
A804-	91 40	STA	(\$40),Y	
A806-	AA	TAX		Verzorg linkpointer naar data
A807-	CA	DEX		sector buffer, weer -256
A808-	68	PLA		bytes voor eerdergenoemde
A809-	48	PHA		buffer.
A80A-	C8	INY		
A80B-	91 40	STA	(\$40),Y	
A80D-	C8	INY		

AB0E-	8A	TXA	
AB0F-	91 40	STA	(\$40),Y
AB11-	CE 63 AA	DEC	\$AA63 Verminder de index-teller
AB14-	F0 17	BEQ	\$AB2D Is dit de laatste buffer ?
AB16-	AA	TAX	Anders moeten we weer
AB17-	68	PLA	pointers gaan opzetten.
AB18-	38	SEC	
AB19-	E9 26	SBC	#\$26 Link komt 38 bytes voor
AB1B-	C8	INY	de data sector buffer.
AB1C-	91 40	STA	(\$40),Y
AB1E-	48	PHA	
AB1F-	8A	TXA	
AB20-	E9 00	SBC	#\$00
AB22-	C8	INY	
AB23-	91 40	STA	(\$40),Y
AB25-	85 41	STA	\$41
AB27-	68	PLA	
AB28-	85 40	STA	\$40
AB2A-	4C E5 A7	JMP	\$A7E5
AB2D-	48	PHA	Het is de laatste buffer
AB2E-	A9 00	LDA	#\$00 dus maken we de link 0
AB30-	C8	INY	
AB31-	91 40	STA	(\$40),Y
AB33-	C8	INY	
AB34-	91 40	STA	(\$40),Y
AB36-	AD B6 AA	LDA	\$AAB6 Welk Basicstype is operationeel?
AB39-	F0 0B	BEQ	\$AB46 Is het Integer ?
AB3B-	68	PLA	
AB3C-	85 74	STA	\$74 Set Applesoft HIMEM
AB3E-	85 70	STA	\$70 en String start pointers.
AB40-	68	PLA	
AB41-	85 73	STA	\$73
AB43-	85 6F	STA	\$6F
AB45-	60	RTS	-----
AB46-	68	PLA	
AB47-	85 4D	STA	\$4D Set Integer HIMEM
AB49-	85 CB	STA	\$CB en Start van programma pointers
AB4B-	68	PLA	
AB4C-	85 4C	STA	\$4C
AB4E-	85 CA	STA	\$CA
AB50-	60	RTS	-----
AB51-	+ A5 39	LDA	\$39 Initialiseer DOS keyboard
AB53-	CD 03 9D	CMF	\$9D03 en video intercept vectors ***** +
AB56-	F0 12	BEQ	\$AB6A Is het keyboard (KSW) gezet ?
AB58-	8D 56 AA	STA	\$AA56 Berg de gevonden vectoren op
AB5B-	A5 38	LDA	\$38
AB5D-	8D 55 AA	STA	\$AA55
AB60-	AD 02 9D	LDA	\$9D02 en vervang ze door DOS' intercept
AB63-	85 38	STA	\$38 routines.
AB65-	AD 03 9D	LDA	\$9D03
AB68-	85 39	STA	\$39
AB6A-	A5 37	LDA	\$37 Is DOS video (CSW) gezet ?
AB6C-	CD 05 9D	CMF	\$9D05
AB6F-	F0 12	BEQ	\$AB83 OK
AB71-	8D 54 AA	STA	\$AA54 Anders ook deze pointers opbergen
AB74-	A5 36	LDA	\$36 en vervangen.
AB76-	8D 53 AA	STA	\$AA53
AB79-	AD 04 9D	LDA	\$9D04
AB7C-	85 36	STA	\$36
AB7E-	AD 05 9D	LDA	\$9D05
AB81-	85 37	STA	\$37
AB83-	60	RTS	-----

+ Aangrijppunt voor "beveiligingen"

A884-	49 4E	EOR	##4E	DOS COMMANDO TABEL *****
A886-	49 D4	EOR	##D4	"INIT" {zie ASCII tabel in Ref Man}
A888-	4C 4F 41	JMP	\$414F	"LOAD"
A888-	C4 53	CPY	\$53	
A88D-	41 56	EOR	(\$56,X)	"SAVE"
A88F-	C5 52	CMP	\$52	
A891-	55 CE	EOR	\$CE,X	"RUN"
A893-	43	???		
A894-	48	PHA		
A895-	41 49	EOR	(\$49,X)	"CHAIN"
A897-	CE 44 45	DEC	\$4544	
A89A-	4C 45 54	JMP	\$5445	"DELETE"
A89D-	C5 4C	CMP	\$4C	"LOCK"
A89F-	4F	???		
A8A0-	43	???		
A8A1-	CB	???		
A8A2-	55 4E	EOR	\$4E,X	
A8A4-	4C 4F 43	JMP	\$434F	
A8A7-	CB	???		"UNLOCK"
A8A8-	43	???		
A8A9-	4C 4F 53	JMP	\$534F	"CLOSE"
A8AC-	C5 52	CMP	\$52	
A8AE-	45 41	EOR	\$41	"READ"
A8B0-	C4 45	CPY	\$45	
A8B2-	58	CLI		
A8B3-	45 C3	EOR	\$C3	"EXEC"
A8B5-	57	???		
A8B6-	52	???		
A8B7-	49 54	EOR	##54	"WRITE"
A8B9-	C5 50	CMP	\$50	
A8BB-	4F	???		
A8BC-	53	???		
A8BD-	49 54	EOR	##54	"POSITION"
A8BF-	49 4F	EOR	##4F	
A8C1-	CE 4F 50	DEC	\$504F	"OPEN"
A8C4-	45 CE	EOR	\$CE	
A8C6-	41 50	EOR	(\$50,X)	"APPEND"
A8C8-	50 45	BVC	\$A90F	
A8CA-	4E C4 52	LSR	\$52C4	
A8CD-	45 4E	EOR	\$4E	"RENAME"
A8CF-	41 4D	EOR	(\$4D,X)	
A8D1-	C5 43	CMP	\$43	
A8D3-	41 54	EOR	(\$54,X)	"CATALOG"
A8D5-	41 4C	EOR	(\$4C,X)	
A8D7-	4F	???		
A8D8-	C7	???		
A8D9-	4D 4F CE	EOR	\$CE4F	"MON"
A8DC-	4E 4F 4D	LSR	\$4D4F	"NOMON"
A8DF-	4F	???		
A8E0-	CE 50 52	DEC	\$5250	"PR//"
A8E3-	A3	???		
A8E4-	49 4E	EOR	##4E	"IN//"
A8E6-	A3	???		
A8E7-	4D 41 58	EOR	\$5841	"MAXFILES"
A8EA-	46 49	LSR	\$49	
A8EC-	4C 45 D3	JMP	\$D345	
A8EF-	46 D0	LSR	\$D0	"FP"
A8F1-	49 4E	EOR	##4E	
A8F3-	D4	???		"INT"
A8F4-	42	???		
A8F5-	53	???		
A8F6-	41 56	EOR	(\$56,X)	"BSAVE"

ABF8-	C5 42	CMP	\$42	
ABFA-	4C 4F 41	JMP	\$414F	"BLOAD"
ABFD-	C4 42	CPY	\$42	
ABFF-	52	???		"BRUN"
A900-	55 CE	EOR	\$CE,X	
A902-	56 45	LSR	\$45,X	"VERIFY"
A904-	52	???		
A905-	49 46	EOR	#\$46	
A907-	D9 00 21	CMP	\$2100,Y	\$A90A begin keyword command
A90A-	70 A0	BVS	\$ABAC	tabel. INIT=2170
A90C-	70 A1	BVS	\$ABAF	LOAD=A070 etc.
A90E-	70 A0	BVS	\$AB80	
A910-	70 20	BVS	\$A932	Deze tabel wordt gebruikt, om
A912-	70 20	BVS	\$A934	de condities van een DOS commando
A914-	70 20	BVS	\$A936	vast te stellen. Het gaat hierbij
A916-	70 20	BVS	\$A938	om een 2 bytes code, waarin 16
A918-	70 60	BVS	\$A97A	conditieflags verscholen gaan.
A91A-	00	BRK		BIT 0 = filenaam optioneel
A91B-	22	???		1 = geen positionele operand
A91C-	06 20	ASL	\$20	2 = filenaam 1 vereist
A91E-	74	???		3 = filenaam 2 vereist
A91F-	22	???		4 = slotnummer vereist
A920-	06 22	ASL	\$22	5 = maxfiles vereist
A922-	04	???		6 = commando binnen programma
A923-	23	???		7 = maak nieuwe file
A924-	78	SEI		8 = CIO conditie
A925-	22	???		9 = V
A926-	70 30	BVS	\$A958	10 = D
A928-	70 40	BVS	\$A96A	11 = S
A92A-	70 40	BVS	\$A96C	12 = L
A92C-	80	???		13 = R
A92D-	40	RTI		14 = B
A92E-	80	???		15 = A parameter optioneel
A92F-	08	PHP		
A930-	00	BRK		
A931-	08	PHP		
A932-	00	BRK		
A933-	04	???		
A934-	00	BRK		
A935-	40	RTI		
A936-	70 40	BVS	\$A97B	
A938-	00	BRK		
A939-	21 79	AND	(\$79,X)	
A93B-	20 71 20	JSR	\$2071	
A93E-	71 20	ADC	(\$20),Y	
A940-	70 D6	BVS	\$A918	\$A941 = "VDSL RBACIO"
A942-	C4 D3	CPY	\$D3	
A944-	CC D2 C2	CPY	\$C2D2	
A947-	C1 C3	CMP	(\$C3,X)	
A949-	C9 CF	CMP	#\$CF	
A94B-	40	RTI		Keyword flag bits V=40
A94C-	20 10 08	JSR	\$0810	D=20, S=10, L=08
A94F-	04	???		R=04
A950-	02	???		B=02
A951-	01 C0	DRA	(\$C0,X)	A=01, C=C0
A953-	A0 90	LDY	#\$90	I=A0, O=90
A955-	00	BRK		Keyword max. omvang tabel
A956-	00	BRK		in standaard volgorde.
A957-	FE 00 01	INC	\$0100,X	A959= min. aantal drives 1
A95A-	00	BRK		
A95B-	02	???		A95B= max. aantal drives 2
A95C-	00	BRK		

A95D-	01 00	ORA	(\$00,X)	
A95F-	07	???		Max. slot=7
A960-	00	BRK		
A961-	01 00	ORA	(\$00,X)	
A963-	FF	???		
A964-	7F	???		Max. L = 32767 of \$7FFF
A965-	00	BRK		Je kunt deze tabel aanpassen.
A966-	00	BRK		
A967-	FF	???		
A968-	7F	???		
A969-	00	BRK		
A96A-	00	BRK		
A96B-	FF	???		
A96C-	7F	???		
A96D-	00	BRK		
A96E-	00	BRK		
A96F-	FF	???		
A970-	FF	???		
A971-	0D 07 8D	ORA	\$8D07	Foutmeldings teksttabel 0=bel rtn
A974-	4C 41 4E	JMP	\$4E41	
A977-	47	???		LANGUAGE NOT AVAILABLE
A978-	55 41	EOR	\$41,X	
A97A-	47	???		
A97B-	45 20	EOR	\$20	
A97D-	4E 4F 54	LSR	\$544F	
A980-	20 41 56	JSR	\$5641	
A983-	41 49	EOR	(\$49,X)	
A985-	4C 41 42	JMP	\$4241	
A988-	4C C5 52	JMP	\$52C5	
A98B-	41 4E	EOR	(\$4E,X)	RANGE ERROR
A98D-	47	???		
A98E-	45 20	EOR	\$20	
A990-	45 52	EOR	\$52	
A992-	52	???		
A993-	4F	???		
A994-	D2	???		
A995-	57	???		
A996-	52	???		WRITE PROTECTED
A997-	49 54	EOR	#\$54	
A999-	45 20	EOR	\$20	
A99B-	50 52	BVC	\$A9EF	
A99D-	4F	???		
A99E-	54	???		
A99F-	45 43	EOR	\$43	
A9A1-	54	???		
A9A2-	45 C4	EOR	\$C4	
A9A4-	45 4E	EOR	\$4E	
A9A6-	44	???		END OF DATA
A9A7-	20 4F 46	JSR	\$464F	
A9AA-	20 44 41	JSR	\$4144	
A9AD-	54	???		
A9AE-	C1 46	CMP	(\$46,X)	
A9B0-	49 4C	EOR	#\$4C	FILE NOT FOUND
A9B2-	45 20	EOR	\$20	
A9B4-	4E 4F 54	LSR	\$544F	
A9B7-	20 46 4F	JSR	\$4F46	
A9BA-	55 4E	EOR	\$4E,X	
A9BC-	C4 56	CPY	\$56	
A9BE-	4F	???		VOLUME MISMATCH
A9BF-	4C 55 4D	JMP	\$4D55	
A9C2-	45 20	EOR	\$20	
A9C4-	4D 49 53	EOR	\$5349	

A9C7-	4D 41 54	EOR	\$5441	
A9CA-	43	???		
A9CB-	C8	INY		
A9CC-	49 2F	EOR	#\$2F	I/O ERROR
A9CE-	4F	???		
A9CF-	20 45 52	JSR	\$5245	
A9D2-	52	???		
A9D3-	4F	???		
A9D4-	D2	???		
A9D5-	44	???		DISK FULL
A9D6-	49 53	EOR	#\$53	
A9D8-	4B	???		
A9D9-	20 46 55	JSR	\$5546	
A9DC-	4C CC 46	JMP	\$46CC	
A9DF-	49 4C	EOR	#\$4C	
A9E1-	45 20	EOR	\$20	FILE LOCKED
A9E3-	4C 4F 43	JMP	\$434F	
A9E6-	4B	???		
A9E7-	45 C4	EOR	\$C4	
A9E9-	53	???		
A9EA-	59 4E 54	EOR	\$544E, Y	SYNTAX ERROR
A9ED-	41 58	EOR	(\$58, X)	
A9EF-	20 45 52	JSR	\$5245	
A9F2-	52	???		
A9F3-	4F	???		
A9F4-	D2	???		
A9F5-	4E 4F 20	LSR	\$204F	
A9F8-	42	???		NO BUFFERS AVAILABLE
A9F9-	55 46	EOR	\$46, X	
A9FB-	46 45	LSR	\$45	
A9FD-	52	???		
A9FE-	53	???		
A9FF-	20 41 56	JSR	\$5641	
AA02-	41 49	EOR	(\$49, X)	
AA04-	4C 41 42	JMP	\$4241	
AA07-	4C C5 46	JMP	\$46C5	
AA0A-	49 4C	EOR	#\$4C	FILE TYPE MISMATCH
AA0C-	45 20	EOR	\$20	
AA0E-	54	???		
AA0F-	59 50 45	EOR	\$4550, Y	
AA12-	20 4D 49	JSR	\$494D	
AA15-	53	???		
AA16-	4D 41 54	EOR	\$5441	
AA19-	43	???		
AA1A-	C8	INY		
AA1B-	50 52	BVC	#\$A6F	PROGRAM TOO LARGE
AA1D-	4F	???		
AA1E-	47	???		
AA1F-	52	???		
AA20-	41 4D	EOR	(\$4D, X)	
AA22-	20 54 4F	JSR	\$4F54	
AA25-	4F	???		
AA26-	20 4C 41	JSR	\$414C	
AA29-	52	???		
AA2A-	47	???		
AA2B-	C5 4E	CMP	\$4E	
AA2D-	4F	???		NOT DIRECT COMMAND
AA2E-	54	???		
AA2F-	20 44 49	JSR	\$4944	
AA32-	52	???		
AA33-	45 43	EOR	\$43	
AA35-	54	???		

AA36-	20 43 4F	JSR	\$4F43	
AA39-	4D 4D 41	EOR	\$414D	
AA3C-	4E C4 8D	LSR	\$8DC4	\$8D = return code
AA3F-	00	BRK		Index offset tabel voor de
AA40-	03	???		foutmeldingen. 1 byte per
AA41-	19 19 24	ORA	\$2419,Y	melding.
AA44-	33	???		
AA45-	3E 4C 5B	ROL	\$5B4C,X	
AA48-	64	???		
AA49-	6D 7B 84	ADC	\$847B	
AA4C-	98	TYA		
AA4D-	AA	TAX		
AA4E-	BB	???		
AA4F-	2D 9B 00	AND	\$009B	DOS VARIABELEN TABEL *****
AA52-	02	???		AA4F=file buffer adres - 2 bytes
AA53-	02	???		AA51=statusflags 00=warmstart
AA54-	C1 1B	CMP	(\$1B,X)	01=read 80=coldstart 40=ASRAM
AA56-	FD 03 03	SBC	\$0303,X	AA52=DOS CSW State code
AA59-	EB	???		AA53=CSW afhandelingsadres 2b
AA5A-	01 00	ORA	(\$01,X)	AA55=KSW afhandelingsadres 2b
AA5C-	A0 06	LDY	\$#06	AA57=Maxfiles
AA5E-	00	BRK		AA59=S,X,Y,A reg save adres 4b
AA5F-	1B	???		AA5D=commando offset
AA60-	71 04	ADC	(\$04),Y	AA5E=MON flags C=40,I=20,O=10
AA62-	00	BRK		AA5F=index laatste commando *2
AA63-	00	BRK		AA60=lengte load of bload
AA64-	01 00	ORA	(\$00,X)	AA64=index vigerend keyword
AA66-	00	BRK		AA65=keyword in instructieregel
AA67-	00	BRK		AA66=volume nummer 2b
AA68-	01 00	ORA	(\$00,X)	AA68=drive nummer 2b
AA6A-	06 00	ASL	\$00	AA6A=slot nummer 2b
AA6C-	00	BRK		AA6C=lengte 2b
AA6D-	00	BRK		AA6E=recordnummer 2b
AA6E-	00	BRK		AA70=byte 2b
AA6F-	00	BRK		AA72=adres 2b
AA70-	00	BRK		AA74=MON code 1 byte
AA71-	00	BRK		
AA72-	00	BRK		*AA60.AA61 AA72.AA73 rtn
AA73-	03	???		geeft L\$ en A\$ recent geladen
AA74-	00	BRK		binary programma.
AA75-	00	BRK		
AA76-	C5 CC	CMP	\$CC	AA75= primaire file naam buffer
AA78-	CC CF A0	CPY	\$A0CF	→ Let er eens op, dat de buffer
AA7B-	A0 A0	LDY	\$#A0	werd vrijgegeven!
AA7D-	A0 A0	LDY	\$#A0	"0ELLO" i.p.v. "HELLO"
AA7F-	A0 A0	LDY	\$#A0	
AA81-	A0 A0	LDY	\$#A0	
AA83-	A0 A0	LDY	\$#A0	
AA85-	A0 A0	LDY	\$#A0	
AA87-	A0 A0	LDY	\$#A0	
AA89-	A0 A0	LDY	\$#A0	
AA8B-	A0 A0	LDY	\$#A0	
AA8D-	A0 A0	LDY	\$#A0	
AA8F-	A0 A0	LDY	\$#A0	
AA91-	A0 A0	LDY	\$#A0	
AA93-	A0 A0	LDY	\$#A0	AA93= secundaire file naam buffer
AA95-	A0 A0	LDY	\$#A0	voor RENAME
AA97-	A0 A0	LDY	\$#A0	
AA99-	A0 A0	LDY	\$#A0	
AA9B-	A0 A0	LDY	\$#A0	
AA9D-	A0 A0	LDY	\$#A0	
AA9F-	A0 A0	LDY	\$#A0	

AAA1-	A0 A0	LDY	##A0	
AAA3-	A0 A0	LDY	##A0	
AAA5-	A0 A0	LDY	##A0	
AAA7-	A0 A0	LDY	##A0	
AAA9-	A0 A0	LDY	##A0	
AAB8-	A0 A0	LDY	##A0	
AAAD-	A0 A0	LDY	##A0	
AAAF-	A0 A0	LDY	##A0	
AAB1-	03	???		AAB1= Maxfiles vervalwaarde (03)
AAB2-	84 00	STY	\$00	AAB2= Ctrl-D
AAB4-	00	BRK		AAB3= EXEC aktief flag 0=niet
AAB5-	00	BRK		AAB4= EXEC file buffer 2 bytes
AAB6-	40	RTI		AAB6= Basic flag: 0=integer
AAB7-	00	BRK		40=AS in ROM, 80=AS RAM
AAB8-	C1 D0	CMP	(\$D0,X)	AAB7= RUN intercept flag
AABA-	D0 CC	BNE	\$AAB8	AAB8= "Applesoft" (ASCII) 9b
AABC-	C5 D3	CMP	\$D3	
AABE-	CF	???		indien eigen programma
AABF-	C6 D4	DEC	\$D4	dan andere naam (AS load)
AAC1-	E8	INX		AAC1= adres RWTs parmlist \$B7E8
AAC2-	B7	???		
AAC3-	BB	???		AAC3= adres VT0C sector buffer
AAC4-	B3	???		
AAC5-	BB	???		AAC5= adres Directory sector buffer
AAC6-	B4 00	LDY	\$00,X	
AAC8-	C0 7E	CPY	##7E	AAC7= laatste DOSbyte + 1! \$C000
AACA-	B3	???		AAC9: File manager subroutine tabel
AACB-	21 AB	AND	(\$AB,X)	Bevat opcodesadresbytes van
AACD-	05 AC	ORA	\$AC	14 file manager functies
AACF-	57	???		b.v. AAD5:adr CATALOG etc.
AAD0-	AC 6F AC	LDY	\$AC6F	
AAD3-	2A	ROL		
AAD4-	AD 97 AD	LDA	\$AD97	
AAD7-	EE AC F5	INC	\$F5AC	
AADA-	AC 39 AC	LDY	\$AC39	
AADD-	11 AD	ORA	(\$AD),Y	
AADF-	8D AE 17	STA	\$17AE	
AAE2-	AD 7E B3	LDA	\$B37E	
AAE5-	7E B3 89	ROR	\$B37E,X	AAE5=READ subcode tabel
AAE8-	AC 95 AC	LDY	\$AC95	AAE7:adr Read 1 byte etc.
AAEB-	86 AC	STX	\$AC	
AAED-	92	???		
AAEE-	AC 7E B3	LDY	\$B37E	
AAF1-	7E B3 BD	ROR	\$BDB3,X	
AAF4-	AC C9 AC	LDY	\$ACC9	AAF1:Write subcode tabel
AAF7-	BA	TSX		
AAF8-	AC C6 AC	LDY	\$ACC6	AAF5=adr write range
AAFB-	7E B3 E0	ROR	\$E0B3,X	
AAFE-	00	BRK		AAFD=extern beginpunt voor
AAFF-	F0 02	BEQ	\$AB03	file manager X=0 ->wijs
AB01-	A2 02	LDX	##02	nieuwe file toe, indien
AB03-	8E 5F AA	STX	\$AA5F	file niet gevonden.
AB06-	BA	TSX		
AB07-	8E 9B B3	STX	\$B39B	File Manager Beginpunt *****
AB0A-	20 6A AE	JSR	\$AE6A	Save S register
AB0D-	AD BB B5	LDA	\$B5BB	herstel file manager werkruimte
AB10-	C9 0D	CMP	##0D	
AB12-	B0 0B	BCS	\$AB1F	Opcode <=13, dan alles OK
AB14-	0A	ASL		Anders volgt foutmelding
AB15-	AA	TAX		Opcode gebruiken als index voor
AB16-	BD CA AA	LDA	\$AACA,X	de f.m. functieroutine.
AB19-	48	PHA		Vector setten. +1

AB1A-	BD C9 AA	LDA	\$AAC9,X
AB1D-	48	PHA	
AB1E-	60	RTS	
AB1F-	4C 63 B3	JMP	\$B363
AB22-	20 28 AB	JSR	\$AB28
AB25-	4C 7F B3	JMP	\$B37F
AB28-	20 DC AB	JSR	\$ABDC
AB2B-	A9 01	LDA	#\$01
AB2D-	8D E3 B5	STA	\$B5E3
AB30-	AE BE B5	LDX	\$B5BE
AB33-	AD BD B5	LDA	\$B5BD
AB36-	D0 05	BNE	\$AB3D
AB38-	E0 00	CPX	#\$00
AB3A-	D0 01	BNE	\$AB3D
AB3C-	E8	INX	
AB3D-	8D E8 B5	STA	\$B5E8
AB40-	8E E9 B5	STX	\$B5E9
AB43-	20 C9 B1	JSR	\$B1C9
AB46-	90 5E	BCC	\$ABA6
AB48-	8E 9C B3	STX	\$B39C
AB4B-	AE 5F AA	LDX	\$AA5F
AB4E-	BD 09 A9	LDA	\$A909,X
AB51-	AE 9C B3	LDX	\$B39C
AB54-	4A	LSR	
AB55-	B0 0D	BCS	\$AB64
AB57-	AD 51 AA	LDA	\$AA51
AB5A-	C9 C0	CMP	#\$C0
AB5C-	D0 03	BNE	\$AB61
AB5E-	4C 5F B3	JMP	\$B35F
AB61-	4C 73 B3	JMP	\$B373
AB64-	A9 00	LDA	#\$00
AB66-	9D E8 B4	STA	\$B4E8,X
AB69-	A9 01	LDA	#\$01
AB6B-	9D E7 B4	STA	\$B4E7,X
AB6E-	8E 9C B3	STX	\$B39C
AB71-	20 44 B2	JSR	\$B244
AB74-	AE 9C B3	LDX	\$B39C
AB77-	9D C7 B4	STA	\$B4C7,X
AB7A-	8D D2 B5	STA	\$B5D2
AB7D-	8D D4 B5	STA	\$B5D4
AB80-	AD F1 B5	LDA	\$B5F1
AB83-	9D C6 B4	STA	\$B4C6,X
AB86-	8D D1 B5	STA	\$B5D1
AB89-	8D D3 B5	STA	\$B5D3
AB8C-	AD C2 B5	LDA	\$B5C2
AB8F-	9D C8 B4	STA	\$B4C8,X
AB92-	20 37 B0	JSR	\$B037
AB95-	20 0C AF	JSR	\$AF0C
AB98-	20 D6 B7	JSR	\$B7D6
AB9B-	20 3A AF	JSR	\$AF3A
AB9E-	AE 9C B3	LDX	\$B39C
ABA1-	A9 06	LDA	#\$06
ABA3-	8D C5 B5	STA	\$B5C5
ABA6-	BD C6 B4	LDA	\$B4C6,X
ABA9-	8D D1 B5	STA	\$B5D1
ABAC-	BD C7 B4	LDA	\$B4C7,X
ABAF-	8D D2 B5	STA	\$B5D2
ABB2-	BD C8 B4	LDA	\$B4C8,X
ABB5-	8D C2 B5	STA	\$B5C2
ABB8-	8D F6 B5	STA	\$B5F6
ABBB-	BD E7 B4	LDA	\$B4E7,X
ABBE-	8D EE B5	STA	\$B5EE

 Naar "RANGE ERROR" opcode 2
 OPEN Afhandeling *****
 Naar exit file manager zonder fout.
 -Open routine, die door veel
 file manager functies wordt benut.
 Set sector lengte op 256 bytes

Indien record lengte 0, dan
 maken we hem 1.

Zoek filenaam in directory
 indien afwezig, dan entry toewijzen.
 Anders door naar ABA6

Controleer de index tabellen

Mag er een nieuwe file worden
 aangelegd ? Dan AB64
 Laden we soms AS van de disk ?
 Nee.
 LANGUAGE NOT AVAILABLE
 FILE NOT FOUND
 Set sectorteller op 1

STX Directory Index
 Wijs sector toe voor T/S List
 Leg T/S List en werkbuffer aan

Schrijf dir sector terug naar disk
 Kies de T/S List buffer
 Maak hem 0
 Schrijf hem terug
 Haal de dir index op
 Set opcode foutmelding op 6
 Zet de "File Not Found"-code weg.
 Copieer T/S List in werkbuffer

Haal T/S op

Copieer file type in werkbuffer

Copieer aantal sectoren /file size

ABC1-	BD E8 B4	LDA	\$B4E8,X	
ABC4-	8D EF B5	STA	\$B5EF	
ABC7-	8E D9 B5	STX	\$B5D9	
ABCA-	A9 FF	LDA	##FF	Set End-of-Data pointer op max.
ABCC-	8D E0 B5	STA	\$B5E0	
ABCF-	8D E1 B5	STA	\$B5E1	
ABD2-	AD E2 B3	LDA	\$B3E2	Aantal entries in T/S List
ABD5-	8D DA B5	STA	\$B5DA	
ABD8-	18	CLC		Lees eerste in, indien aanwezig.
ABD9-	4C 5E AF	JMP	\$AF5E	Lees eerste T/S List sector.
ABDC-	A9 00	LDA	##00	INIT File Manager Werkruimte ****
ABDE-	AA	TAX		
ABDF-	9D D1 B5	STA	\$B5D1,X	
ABE2-	E8	INX		
ABE3-	E0 2D	CPX	##2D	Maak 45 bytes grote ruimte vrij.
ABE5-	D0 F8	BNE	\$ABDF	
ABE7-	AD BF B5	LDA	\$B5BF	Haal volumenummer op
ABEA-	49 FF	EOR	##FF	complementeer het
ABEC-	8D F9 B5	STA	\$B5F9	zet het weg.
ABEF-	AD C0 B5	LDA	\$B5C0	drive nummer
ABF2-	8D F8 B5	STA	\$B5F8	
ABF5-	AD C1 B5	LDA	\$B5C1	slotnummer
ABF8-	0A	ASL		x2
ABF9-	0A	ASL		x4
ABFA-	0A	ASL		x8
ABFB-	0A	ASL		x16
ABFC-	AA	TAX		
ABFD-	8E F7 B5	STX	\$B5F7	zet het in X(*16)weg
AC00-†	A9 11	LDA	##11	Set directory track 17 +
AC02-	8D FA B5	STA	\$B5FA	{ "catalog" }
AC05-	60	RTS		-----
AC06-	20 1D AF	JSR	\$AF1D	CLOSE File afhandeling *****
AC09-	20 34 AF	JSR	\$AF34	Controleer of data cq. IS buffer
AC0C-	20 C3 B2	JSR	\$B2C3	naar disk geschreven moet worden.
AC0F-	A9 02	LDA	##02	Maak onbenutte sectors vrij. B203
AC11-	2D D5 B5	AND	\$B5D5	Kijk of file-omvang is gewijzigd.
AC14-	F0 21	BEQ	\$AC37	Als VTOC ongewijzigd blijft, klaar.
AC16-	20 F7 AF	JSR	\$AFF7	Anders VTOC doorlezen
AC19-	A9 00	LDA	##00	
AC1B-	18	CLC		
AC1C-	20 11 B0	JSR	\$B011	Directory sector doorlezen tot
AC1F-	38	SEC		file is gevonden. Haal index op.
AC20-	CE D8 B5	DEC	\$B5D8	
AC23-	D0 F7	BNE	\$AC1C	Update de sector teller-index
AC25-	AE D9 B5	LDX	\$B5D9	zodat de nieuwe file omvang
AC28-	AD EE B5	LDA	\$B5EE	blijkt.
AC2B-	9D E7 B4	STA	\$B4E7,X	
AC2E-	AD EF B5	LDA	\$B5EF	
AC31-	9D E8 B4	STA	\$B4E8,X	
AC34-	20 37 B0	JSR	\$B037	Schrijf de directorysector terug.
AC37-	4C 7F B3	JMP	\$B37F	Exit file manager.
AC3A-	20 28 AB	JSR	\$AB28	RENAME afhandeling *****
AC3D-	AD F6 B5	LDA	\$B5F6	Welk filetype?
AC40-	30 2B	BMI	\$AC6D	Is file locked ?
AC42-	AD BD B5	LDA	\$B5BD	Nee, dus moet pointer naar
AC45-	85 42	STA	\$42	2de file naam wijzen.
AC47-	AD BE B5	LDA	\$B5BE	
AC4A-	85 43	STA	\$43	
AC4C-	AE 9C B3	LDX	\$B39C	
AC4F-	20 1C B2	JSR	\$B21C	Copiëer nieuwe naam in directory
AC52-	20 37 B0	JSR	\$B037	Schrijf dir terug naar disk
AC55-	4C 7F B3	JMP	\$B37F	Exit file manager

† Je kunt dit veranderen ("beveiligen")
Zie ook AE9F, AEC5 & AEC9 (N/4)

AC5B-	AD BC B5	LDA	\$B5BC	READ afhandelings routine *****
AC5B-	C9 05	CMP	#\$05	Subcode <5 anders
AC5D-	B0 0B	BCS	\$AC6A	volgt een foutmelding
AC5F-	0A	ASL		Gebruik de subcode als index
AC60-	AA	TAX		voor de functiecodetabel
AC61-	BD E6 AA	LDA	\$AAE6,X	t.b.v. Read en
AC64-	4B	PHA		spring naar de juiste afhandelings-
AC65-	BD E5 AA	LDA	\$AAE5,X	routine.
AC68-	4B	PHA		
AC69-	60	RTS		-----
AC6A-	4C 67 B3	JMP	\$B367	Foutafhandeling
AC6D-	4C 7B B3	JMP	\$B37B	File was gelocked
AC70-	AD F6 B5	LDA	\$B5F6	WRITE afhandelingsroutine *****
AC73-	30 F8	BMI	\$AC6D	File was gelocked
AC75-	AD BC B5	LDA	\$B5BC	Controleer de subcode
AC78-	C9 05	CMP	#\$05	Is die kleiner dan 5
AC7A-	B0 EE	BCS	\$AC6A	Is dat zo, dan wordt returncode 3
AC7C-	0A	ASL		Gebruik subcode als index
AC7D-	AA	TAX		voor de write subcodetabel
AC7E-	BD F2 AA	LDA	\$AAF2,X	
AC81-	4B	PHA		
AC82-	BD F1 AA	LDA	\$AAF1,X	Ga naar de juiste afhandelings-
AC85-	4B	PHA		routine.
AC86-	60	RTS		-----
AC87-	20 00 B3	JSR	\$B300	POSITION & READ 1 BYTE *****
AC8A-	20 AB AC	JSR	\$ACAB	Lees volgende byte van file
AC8D-	BD C3 B5	STA	\$B5C3	
AC90-	4C 7F B3	JMP	\$B37F	Exit file manager
AC93-	20 00 B3	JSR	\$B300	POSITION REEKS VAN BYTES ****
AC96-	20 B5 B1	JSR	\$B1B5	Lees byte(s) onder vermindering
AC99-	20 AB AC	JSR	\$ACAB	index
AC9C-	4B	PHA		
AC9D-	20 A2 B1	JSR	\$B1A2	Naar positieroutine
ACA0-	A0 00	LDY	#\$00	
ACA2-	6B	PLA		Bufferadres
ACA3-	91 42	STA	(\$42),Y	
ACA5-	4C 96 AC	JMP	\$AC96	
ACAB-	20 B6 B0	JSR	\$B0B6	Read een data byte ****
ACAB-	B0 0B	BCS	\$ACB8	Zijn we aan het einde van de data?
ACAD-	B1 42	LDA	(\$42),Y	Lees databyte uit buffer
ACAF-	4B	PHA		
ACB0-	20 5B B1	JSR	\$B15B	Tel bij recnummer byte offset 1 op.
ACB3-	20 94 B1	JSR	\$B194	Tel bij positie offset 1 op.
ACB6-	6B	PLA		
ACB7-	60	RTS		-----
ACB8-	4C 6F B3	JMP	\$B36F	Naar End of Data melding
ACBB-	20 00 B3	JSR	\$B300	Position & write 1 byte subcode
ACBE-	AD C3 B5	LDA	\$B5C3	afhandeling.
ACC1-	20 DA AC	JSR	\$ACDA	Schrijf de byte weg
ACC4-	4C 7F B3	JMP	\$B37F	Exit zonder fout.
ACC7-	20 00 B3	JSR	\$B300	Pos & write reeks van bytes.
ACCA-	20 A2 B1	JSR	\$B1A2	Naar positieroutine
ACCD-	A0 00	LDY	#\$00	
ACCF-	B1 42	LDA	(\$42),Y	
ACD1-	20 DA AC	JSR	\$ACDA	Schrijf de byte(s)
ACD4-	20 B5 B1	JSR	\$B1B5	Herbereken data-omvang
ACD7-	4C CA AC	JMP	\$ACCA	
ACDA-	4B	PHA		
ACDB-	20 B6 B0	JSR	\$B0B6	Schrijf byte naar een file ****
ACDE-	6B	PLA		Lees de data sector
ACDF-	91 42	STA	(\$42),Y	Berg de data byte in
ACE1-	A9 40	LDA	#\$40	buffer
				Flag buffer t.b.v. write

ACE3-	0D D5 B5	ORA	\$B5D5
ACE6-	8D D5 B5	STA	\$B5D5
ACE9-	20 5B B1	JSR	\$B15B
ACEC-	4C 94 B1	JMP	\$B194
ACEF-	A9 80	LDA	#\$80
ACF1-	8D 9E B3	STA	\$B39E
ACF4-	D0 05	BNE	\$ACFB
ACF6-	A9 00	LDA	#\$00
ACF8-	8D 9E B3	STA	\$B39E
ACFB-	20 28 AB	JSR	\$AB28
ACFE-	AE 9C B3	LDX	\$B39C
AD01-	BD C8 B4	LDA	\$B4C8,X
AD04-	29 7F	AND	#\$7F
AD06-	0D 9E B3	ORA	\$B39E
AD09-	9D C8 B4	STA	\$B4C8,X
AD0C-	20 37 B0	JSR	\$B037
AD0F-	4C 7F B3	JMP	\$B37F
AD12-	20 00 B3	JSR	\$B300
AD15-	4C 7F B3	JMP	\$B37F
AD18-	20 28 AB	JSR	\$AB28
AD1B-	20 B6 B0	JSR	\$B0B6
AD1E-	B0 EF	BCS	\$AD0F
AD20-	EE E4 B5	INC	\$B5E4
AD23-	D0 F6	BNE	\$AD1B
AD25-	EE E5 B5	INC	\$B5E5
AD28-	4C 1B AD	JMP	\$AD1B
AD2B-	20 28 AB	JSR	\$AB28
AD2E-	AE 9C B3	LDX	\$B39C
AD31-	BD C8 B4	LDA	\$B4C8,X
AD34-	10 03	BPL	\$AD39
AD36-	4C 7B B3	JMP	\$B37B
AD39-	AE 9C B3	LDX	\$B39C
AD3C-	BD C6 B4	LDA	\$B4C6,X
AD3F-	8D D1 B5	STA	\$B5D1
AD42-	9D E6 B4	STA	\$B4E6,X
AD45-	A9 FF	LDA	#\$FF
AD47-	9D C6 B4	STA	\$B4C6,X
AD4A-	BC C7 B4	LDY	\$B4C7,X
AD4D-	8C D2 B5	STY	\$B5D2
AD50-	20 37 B0	JSR	\$B037
AD53-	18	CLC	
AD54-	20 5E AF	JSR	\$AF5E
AD57-	B0 2A	BCS	\$AD83
AD59-	20 0C AF	JSR	\$AF0C
AD5C-	A0 0C	LDY	#\$0C
AD5E-	8C 9C B3	STY	\$B39C
AD61-	B1 42	LDA	(\$42),Y
AD63-	30 0B	BMI	\$AD70
AD65-	F0 09	BEQ	\$AD70
AD67-	48	PHA	
AD68-	C8	INY	
AD69-	B1 42	LDA	(\$42),Y
AD6B-	AB	TAY	
AD6C-	68	PLA	
AD6D-	20 89 AD	JSR	\$AD89
AD70-	AC 9C B3	LDY	\$B39C
AD73-	C8	INY	
AD74-	C8	INY	
AD75-	D0 E7	BNE	\$AD5E
AD77-	AD D3 B5	LDA	\$B5D3
AD7A-	AC D4 B5	LDY	\$B5D4
AD7D-	20 89 AD	JSR	\$AD89

Zet flag weg
Recordnummer-byte offset +1
Exit via positie offset subroutine
LOCK afhandeling *****
Set maskering \$80

UNLOCK afhandeling *****
Set maskering \$00
Waar is de open file ?
Haal index op

Werk filetype byte bij (8x,0x)

Zet filetype weg
Schrijf dir sector weg
Exit file manager
POSITION afhandelingsroutine ***
Call pos routine en exit
VERIFY afhandeling *****
Read data sector van disk
Einde van de file, dan klaar
Anders doorgaan
met lezen

Terug naar af
DELETE afhandelingsroutine *****
Bekijk eerst of file
is gelocked
Nee
Exit met "File Locked" code
Copieer de TS-List pointer (sector)
uit de directory naar de werk-
ruimte en naar de laatste karakter-
byte van de te markeren filenaam
Zet \$FF op de plaats van de
sector-pointer van de TS-List, als
kenmerk voor te wissen file(naam).
Door dit terug te zetten, kan de
file weer teruggehaald worden mits
er (nog) niets overheen gesaved is.
Schrijf dir sector terug B037 en
haal nieuwe sector op. Of klaar?
T/S List buffer
Index T/S paar
Sla track 0 over
Bufferadres
Volgend paar a.u.b.

Werk VTOC bij

Maak de T/S List sector vrij
Volgend paar

Loop terug
Aan eind T/S List
sector vrijmaken en volgende
proberen.

AD80-	38	SEC		
AD81-	B0 D1	BCS	\$AD54	Is er nog een, dan afwerken.
AD83-	20 FB AF	JSR	\$AFFB	Schrijf VTOC bij.
AD86-	4C 7F B3	JMP	\$B37F	Exit.
AD89-	38	SEC		Maak een sector vrij *****
AD8A-	20 DD B2	JSR	\$B2DD	VTOC bit map vrijmaken
AD8D-	A9 00	LDA	#\$00	Maak de sector toewijzing
AD8F-	A2 05	LDX	#\$05	in de werkruimte Ø
AD91-	9D F0 B5	STA	\$B5F0,X	
AD94-	CA	DEX		
AD95-	10 FA	BPL	\$AD91	
AD97-	60	RTS		-----
AD98-	20 DC AB	JSR	\$ABDC	CATALOG functieafhandeling ****
AD9B-	A9 FF	LDA	#\$FF	Maak volumenummer mogelijk
AD9D-	8D F9 B5	STA	\$B5F9	waarbij \$FF=Ø complement
ADA0-	20 F7 AF	JSR	\$AFF7	Waar is de catalog /lees VTOC
ADA3-	A9 16	LDA	#\$16	Geef 22 regels op het beeldscherm
ADA5-	8D 9D B3	STA	\$B39D	Bewaar deze parameter
ADAB-	20 2F AE	JSR	\$AE2F	Sla regel over,
ADAB-	20 2F AE	JSR	\$AE2F	en nog één.
ADAE-	A2 0B	LDX	#\$0B	
ADB0-	BD AF B3	LDA	\$B3AF,X	Print "DISK VOLUME"
ADB3-	20 ED FD	JSR	\$FDED	
ADB6-	CA	DEX		
ADB7-	10 F7	BPL	\$ADB0	
ADB9-	86 45	STX	\$45	Nu het volumenummer
ADBB-	AD F6 B7	LDA	\$B7F6	wegwerken.
ADBE-	85 44	STA	\$44	
ADCO-	20 42 AE	JSR	\$AE42	Print in decimale notatie volnr.
ADC3-	20 2F AE	JSR	\$AE2F	
ADC6-	20 2F AE	JSR	\$AE2F	Skip 2 regels
ADC9-	18	CLC		
ADCA-	20 11 B0	JSR	\$B011	Lees volgende directory sector
ADCD-	B0 5D	BCS	\$AE2C	Bekijk of we klaar zijn.
ADCF-	A2 00	LDX	#\$00	Maak index Ø
ADD1-	BE 9C B3	STX	\$B39C	Is de trackbyte Ø dan zijn we aan
ADD4-	BD C6 B4	LDA	\$B4C6,X	het einde van de directory. Is het
ADD7-	F0 53	BEG	\$AE2C	resultaat Minus, dan is de file
ADD9-	+ 30 4A	BMI	\$AE25	gewist en mag hij niet worden
ADDB-	A0 A0	LDY	#\$A0	afgedrukt! †
ADDD-	BD C8 B4	LDA	\$B4C8,X	Is file "locked"? Filetype
ADE0-	10 02	BPL	\$ADE4	
ADE2-	A0 AA	LDY	#\$AA	"*" - ja, het "lock" teken
ADE4-	98	TYA		Print het.
ADE5-	20 ED FD	JSR	\$FDED	Gebruik filetype als index voor de
ADE8-	BD C8 B4	LDA	\$B4C8,X	filenaam tabel en print naam.
ADEB-	29 7F	AND	#\$7F	Output filetype
ADED-	A0 07	LDY	#\$07	
ADEF-	0A	ASL		
ADFO-	0A	ASL		
ADF1-	B0 03	BCS	\$ADF6	Naar filenaam tabel voor de
ADF3-	88	DEY		type omschrijving van de file
ADF4-	D0 FA	BNE	\$ADFO	
ADF6-	B9 A7 B3	LDA	\$B3A7,Y	Daar komt de naam.
ADF9-	20 ED FD	JSR	\$FDED	Print hem.
ADFC-	A9 A0	LDA	#\$A0	Spatie
ADFE-	20 ED FD	JSR	\$FDED	
AE01-	BD E7 B4	LDA	\$B4E7,X	Nu de file omvang.
AE04-	85 44	STA	\$44	
AE06-	BD E8 B4	LDA	\$B4E8,X	
AE09-	85 45	STA	\$45	
AE0B-	20 42 AE	JSR	\$AE42	

† ADD9: EA EA maakt gedelete files zichtbaar.

AE0E-	A9 A0	LDA	##A0	Spatie
AE10-	20 ED FD	JSR	\$FDED	Print hem.
AE13-	EB	INX		
AE14-	EB	INX		
AE15-	EB	INX		Nu de "programma"/file-naam
AE16-	A0 1D	LDY	##1D	Aantal mogelijke karakters
AE18-	BD C6 B4	LDA	\$B4C6,X	TS track info aflezen
AE1B-	20 ED FD	JSR	\$FDED	Print naam.
AE1E-	EB	INX		
AE1F-	8B	DEY		
AE20-	10 F6	BPL	\$AE18	Hebben we alles ?
AE22-	20 2F AE	JSR	\$AE2F	Sla nu een regel over
AE25-	20 30 B2	JSR	\$B230	Volgende directory entry
AE28-	90 A7	BCC	\$ADD1	Track nummer
AE2A-	B0 9E	BCS	\$ADCA	Lees dir sector
AE2C-	4C 7F B3	JMP	\$B37F	Exit als we klaar zijn.
AE2F-	A9 8D	LDA	##8D	Sla een return; nieuwe regel.
AE31-	20 ED FD	JSR	\$FDED	
AE34-	CE 9D B3	DEC	\$B39D	Regelteller -1
AE37-	D0 0B	BNE	\$AE41	Nog niet 0
AE39-	20 0C FD	JSR	\$FD0C	Wacht op tikje op toetsenbord
AE3C-	A9 15	LDA	##15	Stel de teller weer bij.
AE3E-	8D 9D B3	STA	\$B39D	Nu kunnen we weer 22 regels verder.
AE41-	60	RTS		-----
AE42-	A0 02	LDY	##02	Converteer het getal in \$44
AE44-	A9 00	LDA	##00	tot een drietal decimale cijfers
AE46-	4B	PHA		(integers)
AE47-	A5 44	LDA	\$44	
AE49-	D9 A4 B3	CMP	\$B3A4,Y	
AE4C-	90 12	BCC	\$AE60	
AE4E-	F9 A4 B3	SBC	\$B3A4,Y	
AE51-	85 44	STA	\$44	
AE53-	A5 45	LDA	\$45	
AE55-	E9 00	SBC	##00	
AE57-	85 45	STA	\$45	
AE59-	6B	PLA		
AE5A-	69 00	ADC	##00	
AE5C-	4B	PHA		
AE5D-	4C 47 AE	JMP	\$AE47	
AE60-	6B	PLA		
AE61-	09 B0	ORA	##B0	
AE63-	20 ED FD	JSR	\$FDED	en print dit getal in de Catalog
AE66-	8B	DEY		
AE67-	10 DB	BPL	\$AE44	
AE69-	60	RTS		-----
AE6A-	20 0B AF	JSR	\$AF0B	Kies file manager werkruimte
AE6D-	A0 00	LDY	##00	
AE6F-	8C C5 B5	STY	\$B5C5	return code
AE72-	B1 42	LDA	(\$42),Y	Lees file buffer
AE74-	99 D1 B5	STA	\$B5D1,Y	
AE77-	C8	INY		
AE78-	C0 2D	CPY	##2D	Copiëer 45 bytes uit de file buffer
AE7A-	D0 F6	BNE	\$AE72	terug naar de echte f.m. werkruimte.
AE7C-	1B	CLC		
AE7D-	60	RTS		-----
AE7E-	20 0B AF	JSR	\$AF0B	Save f.m. werkruimte in file buffer
AE81-	A0 00	LDY	##00	
AE83-	B9 D1 B5	LDA	\$B5D1,Y	Lees werkruimte
AE86-	91 42	STA	(\$42),Y	
AE88-	C8	INY		
AE89-	C0 2D	CPY	##2D	Copiëer 45 bytes uit werkruimte
AE8B-	D0 F6	BNE	\$AE83	terug naar de file buffer

AE8D-	60	RTS		
AE8E-	20 DC AB	JSR	\$ABDC	INIT afhandelingsroutine *****
AE91-	A9 04	LDA	#04	Initialiseer file manager werkruimte
AE93-	20 58 B0	JSR	\$B05B	Set command code en call RWTS
AE96-	AD F9 B5	LDA	\$B5F9	Volumenummer
AE99-	49 FF	EOR	##FF	complementeren
AE9B-	8D C1 B3	STA	\$B3C1	
AE9E-	A9 11	LDA	##11	Track die wordt toegewezen
AEA0-	8D EB B3	STA	\$B3EB	
AEA3-	A9 01	LDA	##01	Toewijzing in opgaande richting
AEA5-	8D EC B3	STA	\$B3EC	
AEA8-	A2 38	LDX	##38	Markeer de "mogelijke" 56 tracks
AEA9-	A9 00	LDA	##00	
AEAC-	9D BB B3	STA	\$B3BB,X	
AEAF-	EB	INX		
AE80-	D0 FA	BNE	\$AEAC	
AE82-	A2 0C	LDX	##0C	Maak de VTOC bit map 0, d.w.z.
AE84-	✓ E0 8C	CPX	##8C	maak de 35 Apple tracks "in gebruik".
AE86-	F0 14	BEQ	\$AECC	
AE88-	A0 03	LDY	##03	Sla de drie begin tracks over.
AE8A-	B9 A0 B3	LDA	\$B3A0,Y	
AE8D-	9D F3 B3	STA	\$B3F3,X	✓ Sommige "beveiligde" diskettes hebben 36, zelfs 37 tracks. Wijzig \$8C in 90 en er zijn 36 tracks.
AE80-	EB	INX		Zie voorts B3EF en BEFE!
AE81-	88	DEY		Sla track 44/4=11 over!
AE82-	10 F6	BPL	\$AE8A	
AE84-	† E0 44	CPX	##44	
AE86-	D0 EC	BNE	\$AE84	
AE88-	† A2 48	LDX	##48	Sla track 48/4=12 niet over!
AE8A-	D0 EB	BNE	\$AE84	
AE8C-	20 FB AF	JSR	\$AFFB	
AE8F-	A2 00	LDX	##00	Initialiseer de directory sector buffer
AED1-	8A	TXA		
AED2-	9D BB B4	STA	\$B4BB,X	
AED5-	EB	INX		
AED6-	D0 FA	BNE	\$AED2	
AED8-	20 45 B0	JSR	\$B045	
AEDB-	† A9 11	LDA	##11	Zet directory in track 11
AEDD-	AC F0 B3	LDY	\$B3F0	
AEE0-	88	DEY		
AEE1-	88	DEY		
AEE2-	8D EC B7	STA	\$B7EC	
AEE5-	8D BC B4	STA	\$B4BC	Schrijf de directory nu naar de diskette.
AEE8-	8C BD B4	STY	\$B4BD	
AEEB-	C8	INX		
AEEC-	8C ED B7	STY	\$B7ED	→ † Het is mogelijk de "catalog" naar een andere track te schrijven.
AEEF-	A9 02	LDA	##02	Wijzig de aangegeven adressen en data.
AEF1-	20 58 B0	JSR	\$B05B	Er ontstaat dan een "beveiliging".
AEF4-	AC BD B4	LDY	\$B4BD	Vooral wanneer de directory b.v. op tracks 36 is!
AEF7-	88	DEY		
AEF8-	30 05	BMI	\$AEFF	
AEFA-	D0 EC	BNE	\$AEE8	
AEFC-	98	TYA		
AEFD-	F0 E6	BEQ	\$AEE5	
AEFF-	20 C2 B7	JSR	\$B7C2	Schrijf DOS naar diskette. B7C2/B74A
AF02-	20 4A B7	JSR	\$B74A	Exit zonder foutmelding
AF05-	4C 7F B3	JMP	\$B37F	
AF08-	A2 00	LDX	##00	Kies file manager werk buffer
AF0A-	F0 06	BEQ	\$AF12	
AF0C-	A2 02	LDX	##02	Kies T/S List sector buffer
AF0E-	D0 02	BNE	\$AF12	
AF10-	A2 04	LDX	##04	Kies data buffer
AF12-	BD C7 B5	LDA	\$B5C7,X	

AF15-	85 42	STA	\$42	
AF17-	BD C8 B5	LDA	\$B5C8,X	Werk buffer adres+1
AF1A-	85 43	STA	\$43	
AF1C-	60	RTS		EXIT met \$42 en \$43 geset.
AF1D-	2C D5 B5	BIT	\$B5D5	Controle werkbuffer op data, die naar
AF20-	70 01	BVS	\$AF23	disk geschreven moet worden.
AF22-	60	RTS		
AF23-	20 E4 AF	JSR	\$AFE4	RWTS pointer opzetten
AF26-	A9 02	LDA	#\$02	
AF28-	20 52 B0	JSR	\$B052	Call RWTS voor het schrijven.
AF2B-	A9 BF	LDA	##BF	Reset flag om aan te geven, dat de
AF2D-	2D D5 B5	AND	\$B5D5	data sector gereed is.
AF30-	8D D5 B5	STA	\$B5D5	
AF33-	60	RTS		-----
AF34-	AD D5 B5	LDA	\$B5D5	Controle T/S List buffer op data,
AF37-	30 01	BMI	\$AF3A	die naar disk moet worden geschreven.
AF39-	60	RTS		
AF3A-	20 4B AF	JSR	\$AF4B	RWTS pointer opzetten
AF3D-	A9 02	LDA	#\$02	
AF3F-	20 52 B0	JSR	\$B052	Call RWTS
AF42-	A9 7F	LDA	##7F	Reset flag.
AF44-	2D D5 B5	AND	\$B5D5	
AF47-	8D D5 B5	STA	\$B5D5	
AF4A-	60	RTS		-----
AF4B-	AD C9 B5	LDA	\$B5C9	IOB gereed maken voor RWTS Call
AF4E-	8D F0 B7	STA	\$B7F0	Copieer adres T/S List buffer
AF51-	AD CA B5	LDA	\$B5CA	in parmlist
AF54-	8D F1 B7	STA	\$B7F1	
AF57-	AE D3 B5	LDX	\$B5D3	T/S van sector
AF5A-	AC D4 B5	LDY	\$B5D4	
AF5D-	60	RTS		-----
AF5E-	08	PHP		Lees de T/S List in file buffer in.
AF5F-	20 34 AF	JSR	\$AF34	C=0 : eerste T/S sector gewenst
AF62-	20 4B AF	JSR	\$AF4B	C=1 ; volgende T/S sector gewenst.
AF65-	20 0C AF	JSR	\$AF0C	Kies de T/S List buffer
AF68-	2B	PLP		Is carry flag "clear", dan eerste
AF69-	B0 09	BCS	\$AF74	T/S List sector.
AF6B-	AE D1 B5	LDX	\$B5D1	
AF6E-	AC D2 B5	LDY	\$B5D2	Haal T/S Link op
AF71-	4C B5 AF	JMP	\$AFB5	
AF74-	A0 01	LDY	##01	Is carry flag geset, lees dan
AF76-	B1 42	LDA	(\$42),Y	volgende T/S List sector
AF78-	F0 08	BEQ	\$AFB2	Is de volgende track 0, dan fout.
AF7A-	AA	TAX		Is er nog een T/S List aanwezig,
AF7B-	CB	INY		dan inlezen.
AF7C-	B1 42	LDA	(\$42),Y	
AF7E-	AB	TAY		
AF7F-	4C B5 AF	JMP	\$AFB5	Set RWTS code
AF82-	AD BB B5	LDA	\$B5BB	Is de file manager opcode 04 (write),
AF85-	C9 04	CMP	##04	dan zal geen foutmelding optreden.
AF87-	F0 02	BEQ	\$AFBB	
AF89-	38	SEC		
AF8A-	60	RTS		
AF8B-	20 44 B2	JSR	\$B244	Wijs een nieuwe sector toe.
AF8E-	A0 02	LDY	##02	Verwijzing oude T/S List sector
AF90-	91 42	STA	(\$42),Y	naar nieuwe T/S List track/sector
AF92-	48	PHA		
AF93-	8B	DEY		
AF94-	AD F1 B5	LDA	\$B5F1	
AF97-	91 42	STA	(\$42),Y	
AF99-	48	PHA		
AF9A-	20 3A AF	JSR	\$AF3A	

AF9D-	20 D6 B7	JSR	\$B7D6	Maak buffer vrij voor nieuwe T/SL
AFA0-	A0 05	LDY	##05	Bereken het relatieve sectornummer
AFA2-	AD DE B5	LDA	\$B5DE	van de eerste sector op +5,+6 in
AFA5-	91 42	STA	(\$42),Y	buffer.
AFA7-	C8	INY		
AFA8-	AD DF B5	LDA	\$B5DF	
AFAB-	91 42	STA	(\$42),Y	
AFAD-	68	PLA		
AFAE-	AA	TAX		
AFAF-	68	PLA		
AFB0-	AB	TAY		
AFB1-	A9 02	LDA	##02	Set opcode voor het schrijven van
AFB3-	D0 02	BNE	\$AFB7	de nieuwe TS/list
AFB5-	A9 01	LDA	##01	Is acc=1 dan lees oude TSL
AFB7-	8E D3 B5	STX	\$B5D3	Is acc=2 dan schrijf nieuwe sector
AFBA-	8C D4 B5	STY	\$B5D4	
AFBD-	20 52 B0	JSR	\$B052	
AFC0-	A0 05	LDY	##05	Bereken het relatieve sectornummer
AFC2-	B1 42	LDA	(\$42),Y	van de laatste sector uit de TSL
AFC4-	8D DC B5	STA	\$B5DC	en zet die in de werkruimte.
AFC7-	18	CLC		
AFC8-	6D DA B5	ADC	\$B5DA	
AFCB-	8D DE B5	STA	\$B5DE	
AFCE-	C8	INY		
AFCF-	B1 42	LDA	(\$42),Y	
AFD1-	8D DD B5	STA	\$B5DD	
AFD4-	6D DB B5	ADC	\$B5DB	
AFD7-	8D DF B5	STA	\$B5DF	
AFDA-	18	CLC		
AFDB-	60	RTS		-----
AFDC-	20 E4 AF	JSR	\$AFE4	Lees een data sector
AFDF-	A9 01	LDA	##01	
AFE1-	4C 52 B0	JMP	\$B052	
AFE4-	AC CB B5	LDY	\$B5CB	Copiëer adres data sector buffer
AFE7-	AD CC B5	LDA	\$B5CC	naar RWTS parmlist
AFEA-	8C F0 B7	STY	\$B7F0	
AFED-	8D F1 B7	STA	\$B7F1	
AFF0-	AE D6 B5	LDX	\$B5D6	Haal track en sector op.
AFF3-	AC D7 B5	LDY	\$B5D7	
AFF6-	60	RTS		-----
AFF7-	A9 01	LDA	##01	Lees of schrijf VTOC buffer
AFF9-	D0 02	BNE	\$AFFD	Lees VTOC
AFFB-	A9 02	LDA	##02	Schrijf VTOC
AFFD-	AC C3 AA	LDY	\$AAC3	
B000-	8C F0 B7	STY	\$B7F0	Copiëer VTOC sector buffer adres
B003-	AC C4 AA	LDY	\$AAC4	in parmlist van RWTS
B006-	8C F1 B7	STY	\$B7F1	
B009-	AE FA B5	LDX	\$B5FA	Haal track/sectornummer op
B00C-	A0 00	LDY	##00	en gebruik sector 0
B00E-	4C 52 B0	JMP	\$B052	Exit via RWTS driver.
B011-	08	PHP		
B012-	20 45 B0	JSR	\$B045	Lees een directory sector *****
B015-	28	PLP		Code in carry flag l=volgende.
B016-	B0 08	BCS	\$B020	
B018-	AC BD B3	LDY	\$B3BD	Lees eerste dir T/S uit VTOC
B01B-	AE BC B3	LDX	\$B3BC	met offset +1, +2
B01E-	D0 0A	BNE	\$B02A	
B020-	AE BC B4	LDX	\$B4BC	Lees T/S uit dir sector met offset
B023-	D0 02	BNE	\$B027	+1, +2. Is de track 0 dan fout.
B025-	38	SEC		Carry (foutmelding) set.
B026-	60	RTS		-----
B027-	AC BD B4	LDY	\$B4BD	Lees nu volgende dir sector

B02A-	BE 97 B3	STX	\$B397	
B02D-	8C 98 B3	STY	\$B398	
B030-	A9 01	LDA	#\$01	
B032-	20 52 B0	JSR	\$B052	RWTS driver voor lezen van sector.
B035-	18	CLC		
B036-	60	RTS		-----
B037-	20 45 B0	JSR	\$B045	Schrijf directory sector.
B03A-	AE 97 B3	LDX	\$B397	Set buffer pointers.
B03D-	AC 98 B3	LDY	\$B398	
B040-	A9 02	LDA	#\$02	
B042-	4C 52 B0	JMP	\$B052	RWTS driver voor schrijven sector.
B045-	AD C5 AA	LDA	\$AAC5	Maak RWTS gereed voor het lezen en
B048-	8D F0 B7	STA	\$B7F0	schrijven van de direct. buffer ****
B04B-	AD C6 AA	LDA	\$AAC6	Copiëer buffer adres in parmlist
B04E-	8D F1 B7	STA	\$B7F1	
B051-	60	RTS		-----
B052-	8E EC B7	STX	\$B7EC	RWTS DRIVER *****
B055-	8C ED B7	STY	\$B7ED	Set T/S in RWTS parmlist
B058-	8D F4 B7	STA	\$B7F4	Set commando code (lezen, schrijven)
B05B-	C9 02	CMP	#\$02	
B05D-	D0 06	BNE	\$B065	Indien aan het schrijven,
B05F-	0D D5 B5	ORA	\$B5D5	dan juiste flag setten.
B062-	8D D5 B5	STA	\$B5D5	
B065-	AD F9 B5	LDA	\$B5F9	Complementeer het volumenummer
B068-	49 FF	EOR	#\$FF	
B06A-	8D EB B7	STA	\$B7EB	
B06D-	AD F7 B5	LDA	\$B5F7	Initialiseer slotnummer x16
B070-	8D E9 B7	STA	\$B7E9	
B073-	AD F8 B5	LDA	\$B5F8	Init drive nummer
B076-	8D EA B7	STA	\$B7EA	
B079-	AD E2 B5	LDA	\$B5E2	Init sectorlengte en len+1
B07C-	8D F2 B7	STA	\$B7F2	
B07F-	AD E3 B5	LDA	\$B5E3	B085 Settype (\$01) in parmlist
B082-	8D F3 B7	STA	\$B7F3	
B085-	A9 01	LDA	#\$01	
B087-	8D E8 B7	STA	\$B7E8	IOB type
B08A-	AC C1 AA	LDY	\$AAC1	Call RWTS met parmlist pointer
B08D-	AD C2 AA	LDA	\$AAC2	
B090-	20 B5 B7	JSR	\$B7B5	
B093-	AD F6 B7	LDA	\$B7F6	Volumenummer testen en
B096-	8D BF B5	STA	\$B5BF	copiëren in parmlist
B099-	A9 FF	LDA	#\$FF	
B09B-	8D EB B7	STA	\$B7EB	Verwacht volumenummer
B09E-	B0 01	BCS	\$B0A1	Niet goed, geld weg ...
B0A0-	60	RTS		-----
B0A1-	AD F5 B7	LDA	\$B7F5	Converteer RWTS foutcode tot
B0A4-	A0 07	LDY	#\$07	code, die door de file manager
B0A6-	C9 20	CMP	#\$20	kan worden gebruikt.
B0A8-	F0 08	BEQ	\$B0B2	07=VOLUME MISMATCH ERROR
B0AA-	A0 04	LDY	#\$04	WRITE PROTECTED
B0AC-	C9 10	CMP	#\$10	
B0AE-	F0 02	BEQ	\$B0B2	
B0B0-	A0 08	LDY	#\$08	I/O ERROR (alle overige fouten)
B0B2-	98	TYA		
B0B3-	4C B5 B3	JMP	\$B3B5	EXIT via foutafhandeling.
B0B6-	AD E4 B5	LDA	\$B5E4	Lees volgende data sector (eventueel)
B0B9-	CD E0 B5	CMP	\$B5E0	Is de geldende file positie binnen
B0BC-	D0 08	BNE	\$B0C6	de data sector in het geheugen ?
B0BE-	AD E5 B5	LDA	\$B5E5	
B0C1-	CD E1 B5	CMP	\$B5E1	
B0C4-	F0 66	BEQ	\$B12C	Als dat zo is, dan B12C
B0C6-	20 1D AF	JSR	\$AF1D	Moet sector naar disk worden geschre-
				ven ?

BOC9-	AD E5 B5	LDA	\$B5E5	Is de file positie binnen de omvang van sectoren, die gevonden wordt in de TSL, die in het geheugen is ?
BOCC-	CD DD B5	CMP	\$B5DD	
BOCF-	90 1C	BCC	\$B0ED	
BOD1-	DO 08	BNE	\$B0DB	
BOD3-	AD E4 B5	LDA	\$B5E4	
BOD6-	CD DC B5	CMP	\$B5DC	
BOD9-	90 12	BCC	\$B0ED	
BODB-	AD E5 B5	LDA	\$B5E5	
BODE-	CD DF B5	CMP	\$B5DF	
BOE1-	90 10	BCC	\$B0F3	Nee.
BOE3-	DO 08	BNE	\$B0ED	Lees elke TSL na en zoek de juiste.
BOE5-	AD E4 B5	LDA	\$B5E4	
BOEB-	CD DE B5	CMP	\$B5DE	
BOEB-	90 06	BCC	\$B0F3	
BOED-	20 5E AF	JSR	\$AF5E	Lees nieuwe TSL in, carry set, of cleared.
BOFO-	90 D7	BCC	\$B0C9	-----
BOF2-	60	RTS		
BOF3-	38	SEC		Maak entry in deze TSL
BOF4-	AD E4 B5	LDA	\$B5E4	Bereken de filepositie.
BOF7-	ED DC B5	SBC	\$B5DC	
BOFA-	0A	ASL		
BOFB-	69 0C	ADC	\$#0C	
BOFD-	AB	TAY		
BOFE-	20 0C AF	JSR	\$AF0C	Kies de TSL buffer
B101-	B1 42	LDA	(\$42),Y	Welke track voor data sector ?
B103-	DO 0F	BNE	\$B114	Ongelijk 0 ?
B105-	AD BB B5	LDA	\$B5BB	Anders ontstaat een fout, als er nu niet wordt geschreven.
B108-	C9 04	CMP	\$#04	
B10A-	F0 02	BEQ	\$B10E	
B10C-	38	SEC		
B10D-	60	RTS		-----
B10E-	20 34 B1	JSR	\$B134	Wijs een nieuwe sector toe,
B111-	4C 20 B1	JMP	\$B120	en werk de administratie bij.
B114-	8D D6 B5	STA	\$B5D6	Lees data aan de hand van bestaande TSL
B117-	CB	INY		
B118-	B1 42	LDA	(\$42),Y	
B11A-	8D D7 B5	STA	\$B5D7	
B11D-	20 DC AF	JSR	\$AFDC	
B120-	AD E4 B5	LDA	\$B5E4	Laadt het sectornummer, dat het laatst werd gelezen op. Zet het weg in werkruimte.
B123-	8D E0 B5	STA	\$B5E0	
B126-	AD E5 B5	LDA	\$B5E5	
B129-	8D E1 B5	STA	\$B5E1	
B12C-	20 10 AF	JSR	\$AF10	Kies een data buffer
B12F-	AC E6 B5	LDY	\$B5E6	
B132-	18	CLC		
B133-	60	RTS		-----
B134-	8C 9D B3	STY	\$B39D	Voeg een nieuwe data sector aan de file toe.
B137-	20 44 B2	JSR	\$B244	Zet TS nummers in TSL entry
B13A-	AC 9D B3	LDY	\$B39D	
B13D-	CB	INY		
B13E-	91 42	STA	(\$42),Y	
B140-	8D D7 B5	STA	\$B5D7	
B143-	88	DEY		
B144-	AD F1 B5	LDA	\$B5F1	
B147-	91 42	STA	(\$42),Y	
B149-	8D D6 B5	STA	\$B5D6	
B14C-	20 10 AF	JSR	\$AF10	Kies een data buffer en schoon die op.
B14F-	20 D6 B7	JSR	\$B7D6	Set flags in de schrijf-mode.
B152-	A9 C0	LDA	\$#C0	
B154-	0D D5 B5	ORA	\$B5D5	
B157-	8D D5 B5	STA	\$B5D5	
B15A-	60	RTS		-----

B15B-	AE EA B5	LDX	\$B5EA	Werk recordnummer bij van de file
B15E-	8E BD B5	STX	\$B5BD	
B161-	AE EB B5	LDX	\$B5EB	
B164-	8E BE B5	STX	\$B5BE	
B167-	AE EC B5	LDX	\$B5EC	Werk byte offset bij
B16A-	AC ED B5	LDY	\$B5ED	
B16D-	8E BF B5	STX	\$B5BF	
B170-	8C C0 B5	STY	\$B5C0	
B173-	EB	INX		Diverse hulproutines voor de
B174-	DO 01	BNE	\$B177	filebehandeling.
B176-	CB	INY		
B177-	CC E9 B5	CPY	\$B5E9	
B17A-	DO 11	BNE	\$B18D	
B17C-	EC EB B5	CPX	\$B5EB	
B17F-	DO 0C	BNE	\$B18D	
B181-	A2 00	LDX	\$*00	Is de byte offset gelijk aan de
B183-	A0 00	LDY	\$*00	recordlengte, dan de byte offset
B185-	EE EA B5	INC	\$B5EA	op 0 zetten
B188-	DO 03	BNE	\$B18D	
B18A-	EE EB B5	INC	\$B5EB	
B18D-	8E EC B5	STX	\$B5EC	
B190-	8C ED B5	STY	\$B5ED	
B193-	60	RTS		
B194-	EE E6 B5	INC	\$B5E6	File positie offset bijwerken.
B197-	DO 08	BNE	\$B1A1	
B199-	EE E4 B5	INC	\$B5E4	
B19C-	DO 03	BNE	\$B1A1	
B19E-	EE E5 B5	INC	\$B5E5	
B1A1-	60	RTS		-----
B1A2-	AC C3 B5	LDY	\$B5C3	Copiëer omvangstabel
B1A5-	AE C4 B5	LDX	\$B5C4	
B1A8-	84 42	STY	\$42	naar \$42
B1AA-	86 43	STX	\$43	\$43
B1AC-	EE C3 B5	INC	\$B5C3	Increment range adres in parmlist
B1AF-	DO 03	BNE	\$B1B4	
B1B1-	EE C4 B5	INC	\$B5C4	
B1B4-	60	RTS		-----
B1B5-	AC C1 B5	LDY	\$B5C1	Verminder de omvangstabel
B1B8-	DO 08	BNE	\$B1C2	
B1BA-	AE C2 B5	LDX	\$B5C2	
B1BD-	F0 07	BEQ	\$B1C6	
B1BF-	CE C2 B5	DEC	\$B5C2	Filetype
B1C2-	CE C1 B5	DEC	\$B5C1	Slot
B1C5-	60	RTS		-----
B1C6-	4C 7F B3	JMP	\$B37F	Exit file manager
B1C9-	20 F7 AF	JSR	\$AFF7	Zoek filenaam in directory ***
B1CC-	AD C3 B5	LDA	\$B5C3	Lees VTOC en set pointers naar
B1CF-	B5 42	STA	\$42	filenaam, die wordt gezocht.
B1D1-	AD C4 B5	LDA	\$B5C4	
B1D4-	B5 43	STA	\$43	
B1D6-	A9 01	LDA	\$*01	We lezen de directory 2 keer.
B1D8-	8D 9D B3	STA	\$B39D	
B1DB-	A9 00	LDA	\$*00	
B1DD-	8D DB B5	STA	\$B5DB	Index VTOC t.b.v. dir track
B1EO-	18	CLC		
B1E1-	EE DB B5	INC	\$B5DB	Lees de dir sector in.
B1E4-	20 11 B0	JSR	\$B011	
B1E7-	B0 51	BCS	\$B23A	
B1E9-	A2 00	LDX	\$*00	Controleer einde directory (0)
B1EB-	8E 9C B3	STX	\$B39C	of gewiste file (minus)
B1EE-	BD C6 B4	LDA	\$B4C6,x	
B1F1-	F0 1F	BEQ	\$B212	

B1F3-	30 22	BMI	\$B217	
B1F5-	A0 00	LDY	#\$00	
B1F7-	EB	INX		
B1F8-	EB	INX		
B1F9-	EB	INX		Vergelijk de filenaam met
B1FA-	B1 42	LDA	(\$42),Y	de directory entry
B1FC-	DD C6 B4	CMP	\$B4C6,X	
B1FF-	DO 0A	BNE	\$B20B	
B201-	C8	INY		
B202-	C0 1E	CPY	#\$1E	
B204-	DO F3	BNE	\$B1F9	
B206-	AE 9C B3	LDX	\$B39C	Klopt het allemaal ?
B209-	18	CLC		Dan carry clear en index in X
B20A-	60	RTS		-----
B20B-	20 30 B2	JSR	\$B230	Klopt het niet, dan index ophogen
B20E-	90 DB	BCC	\$B1EB	en terug.
B210-	B0 CF	BCS	\$B1E1	Einde sector, dan volgende.
B212-	AC 9D B3	LDY	\$B39D	Is dit pass 1 ?
B215-	DO C1	BNE	\$B1D8	
B217-	AC 9D B3	LDY	\$B39D	Is dit pass 1, dan entry overslaan
B21A-	DO EF	BNE	\$B20B	of entry toewijzen.
B21C-	A0 00	LDY	#\$00	Copiëer filenaam in directory
B21E-	EB	INX		entry
B21F-	EB	INX		
B220-	EB	INX		
B221-	B1 42	LDA	(\$42),Y	
B223-	9D C6 B4	STA	\$B4C6,X	
B226-	C8	INY		
B227-	C0 1E	CPY	#\$1E	
B229-	DO F5	BNE	\$B220	
B22B-	AE 9C B3	LDX	\$B39C	
B22E-	38	SEC		
B22F-	60	RTS		-----
B230-	18	CLC		Wij zullen doorgaan ...
B231-	AD 9C B3	LDA	\$B39C	Volgende directory entry
B234-	69 23	ADC	#\$23	35 is entry lengte (catalog)
B236-	AA	TAX		
B237-	E0 F5	CPX	#\$F5	
B239-	60	RTS		-----
B23A-	A9 00	LDA	#\$00	Tweede pass in afzoeken dir
B23C-	AC 9D B3	LDY	\$B39D	
B23F-	DO 97	BNE	\$B1D8	
B241-	4C 77 B3	JMP	\$B377	Verdorie: "DISK FULL"
B244-	AD F1 B5	LDA	\$B5F1	Disk sector toewijzingsroutine ****
B247-	F0 21	BEQ	\$B26A	Is er al een track/sector toegewe-
B249-	CE F0 B5	DEC	\$B5F0	zen ? Anders zoeken naar plaats.
B24C-	30 17	BMI	\$B265	
B24E-	18	CLC		Gevonden! Nu de bitmap aanpassen.
B24F-	A2 04	LDX	#\$04	
B251-	3E F1 B5	ROL	\$B5F1,X	
B254-	CA	DEX		
B255-	DO FA	BNE	\$B251	
B257-	90 F0	BCC	\$B249	Sector blijkt in gebruik.
B259-	EE EE B5	INC	\$B5EE	Sector is vrij - filesize +1
B25C-	DO 03	BNE	\$B261	
B25E-	EE EF B5	INC	\$B5EF	
B261-	AD F0 B5	LDA	\$B5F0	
B264-	60	RTS		-----
B265-	A9 00	LDA	#\$00	Wordt track niet gebruikt ?
B267-	8D F1 B5	STA	\$B5F1	
B26A-	A9 00	LDA	#\$00	Reset flag t.b.v. afzoeken lege
B26C-	8D 9E B3	STA	\$B39E	sectoren.

B26F-	20 F7 AF	JSR	\$AFF7	RWVTOC
B272-	18	CLC		Ga naar laatste track, die beschreven
B273-	AD EB B3	LDA	\$B3EB	werd en voeg de richtings-offset toe
B276-	6D EC B3	ADC	\$B3EC	en bekijk of we nu op track 0 zijn.
B279-	F0 09	BEQ	\$B284	
B27B-	CD EF B3	CMP	\$B3EF	
B27E-	90 14	BCC	\$B294	
B280-	A9 FF	LDA	#\$FF	Zijn we voorbij track 34 ?
B282-	D0 0A	BNE	\$B28E	Dan van richting veranderen.
B284-	AD 9E B3	LDA	\$B39E	
B287-	D0 37	BNE	\$B2C0	Dan kan de diskette vol zijn.
B289-	A9 01	LDA	#\$01	
B28B-	8D 9E B3	STA	\$B39E	
B28E-	8D EC B3	STA	\$B3EC	Is het dan de 2e keer, dat we op
B291-	18	CLC		track 0 zijn ?
B292-†	69 11	ADC	#\$11	Begin op track 11 en tel af.
B294-	8D EB B3	STA	\$B3EB	
B297-	8D F1 B5	STA	\$B5F1	
B29A-	AB	TAY		
B29B-	0A	ASL		
B29C-	0A	ASL		
B29D-	AB	TAY		
B29E-	A2 04	LDX	#\$04	Bereken de bitmap index
B2A0-	18	CLC		(tracknummer *4)
B2A1-	B9 F6 B3	LDA	\$B3F6,Y	Copieer de track bit map in de
B2A4-	9D F1 B5	STA	\$B5F1,X	werkruimte, om te zien of de
B2A7-	F0 06	BEQ	\$B2AF	track vol is - 4 * 0
B2A9-	38	SEC		
B2AA-	A9 00	LDA	#\$00	
B2AC-	99 F6 B3	STA	\$B3F6,Y	In ieder geval worden alle bytes
B2AF-	88	DEY		in de VTOC op 0 gezet zodat daaruit
B2B0-	CA	DEX		sector toewijzing blijkt.
B2B1-	D0 EE	BNE	\$B2A1	
B2B3-	90 BD	BCC	\$B272	Volgende track s.v.p.
B2B5-	20 FB AF	JSR	\$AFFB	Schrijf VTOC naar diskette
B2B8-	AD F0 B3	LDA	\$B3F0	
B2BB-	8D F0 B5	STA	\$B5F0	Sectornummer nu laatste sector
B2BE-	D0 89	BNE	\$B249	in track. Wijs vrije sector toe.
B2C0-	4C 77 B3	JMP	\$B377	Error: Disk Full
B2C3-	AD F1 B5	LDA	\$B5F1	Geef nu nog niet benutte sectoren
B2C6-	D0 01	BNE	\$B2C9	vrij. Vond toewijzing plaats ?
B2C8-	60	RTS		-----
B2C9-	48	PHA		
B2CA-	20 F7 AF	JSR	\$AFF7	Werk VTOC bij.
B2CD-	AC F0 B5	LDY	\$B5F0	
B2D0-	68	PLA		
B2D1-	18	CLC		
B2D2-	20 DD B2	JSR	\$B2DD	Roteer de gehele bit mask.
B2D5-	A9 00	LDA	#\$00	Voeg hem in de VTOC bit map.
B2D7-	8D F1 B5	STA	\$B5F1	En check eventuele vrije sectoren.
B2DA-	4C FB AF	JMP	\$AFFB	Op deze wijze verkrijgen wij de
B2DD-	A2 FC	LDX	#\$FC	track, die wij zouden kunnen
B2DF-	7E F6 B4	ROR	\$B4F6,X	benutten.
B2E2-	EB	INX		
B2E3-	D0 FA	BNE	\$B2DF	
B2E5-	C8	INY		
B2E6-	CC F0 B3	CPY	\$B3F0	
B2E9-	D0 F2	BNE	\$B2DD	
B2EB-	0A	ASL		
B2EC-	0A	ASL		
B2ED-	AB	TAY		
B2EE-	F0 0F	BEQ	\$B2FF	

† indien catalog elders, dan wijzigen.

B2F0-	A2 04	LDX	##04	
B2F2-	BD F1 B5	LDA	\$B5F1,X	
B2F5-	19 F6 B3	ORA	\$B3F6,Y	Voeg de file manager bits toe aan
B2FB-	99 F6 B3	STA	\$B3F6,Y	de VTOC en geef daarmee sectoren
B2FB-	88	DEY		vrij, die niet door de file worden
B2FC-	CA	DEX		gebruikt.
B2FD-	D0 F3	BNE	\$B2F2	
B2FF-	60	RTS		
B300-	AD BD B5	LDA	\$B5BD	Bereken de file positie.
B303-	8D E6 B5	STA	\$B5E6	Zet recordnummer in parmlist in
B306-	8D EA B5	STA	\$B5EA	werkruimte en in sector offsets.
B309-	AD BE B5	LDA	\$B5BE	
B30C-	8D E4 B5	STA	\$B5E4	
B30F-	8D EB B5	STA	\$B5EB	
B312-	A9 00	LDA	##00	
B314-	8D E5 B5	STA	\$B5E5	
B317-	A0 10	LDY	##10	
B319-	AA	TAX		
B31A-	AD E6 B5	LDA	\$B5E6	Voer een 16 bit vermenigvuldiging
B31D-	4A	LSR		uit, na de hi-bits van de sector-
B31E-	B0 03	BCS	\$B323	offset te hebben afgezet.
B320-	8A	TXA		
B321-	90 0E	BCC	\$B331	
B323-	18	CLC		
B324-	AD E5 B5	LDA	\$B5E5	
B327-	6D E8 B5	ADC	\$B5E8	
B32A-	8D E5 B5	STA	\$B5E5	
B32D-	8A	TXA		
B32E-	6D E9 B5	ADC	\$B5E9	File positie=3 bytes (recnum
B331-	6A	ROR		x reclen)
B332-	6E E5 B5	ROR	\$B5E5	Hier komen de antwoorden.
B335-	6E E4 B5	ROR	\$B5E4	
B338-	6E E6 B5	ROR	\$B5E6	
B33B-	88	DEY		
B33C-	D0 DB	BNE	\$B319	
B33E-	AD BF B5	LDA	\$B5BF	
B341-	8D EC B5	STA	\$B5EC	
B344-	6D E6 B5	ADC	\$B5E6	
B347-	8D E6 B5	STA	\$B5E6	Voeg de byte offset uit de parmlist
B34A-	AD C0 B5	LDA	\$B5C0	toe aan de 3 byte positiewaarde.
B34D-	8D ED B5	STA	\$B5ED	
B350-	6D E4 B5	ADC	\$B5E4	
B353-	8D E4 B5	STA	\$B5E4	
B356-	A9 00	LDA	##00	
B358-	6D E5 B5	ADC	\$B5E5	
B35B-	8D E5 B5	STA	\$B5E5	
B35E-	60	RTS		
B35F-	A9 01	LDA	##01	-----
B361-	D0 22	BNE	\$B385	Foutmeldingscodes.
B363-	A9 02	LDA	##02	Language not available
B365-	D0 1E	BNE	\$B385	
B367-	A9 03	LDA	##03	Range error
B369-	D0 1A	BNE	\$B385	
B36B-	A9 04	LDA	##04	Range error (subcode)
B36D-	D0 16	BNE	\$B385	
B36F-	A9 05	LDA	##05	Write protected
B371-	D0 12	BNE	\$B385	
B373-	A9 06	LDA	##06	End of data
B375-	D0 0E	BNE	\$B385	
B377-	4C ED BF	JMP	\$BFED	File not found
B37A-	EA	NOP		Disk full en alle files sluiten.
B37B-	A9 0A	LDA	##0A	

B37D-	D0 06	BNE	\$B385	File Locked
B37F-	AD C5 B5	LDA	\$B5C5	No error - returncode
B382-	18	CLC		
B383-	90 01	BCC	\$B386	Clear carry flag.
B385-	38	SEC		Standaard foutafhandelingsroutine.
B386-	08	PHP		Er is een fout. C-flag
B387-	8D C5 B5	STA	\$B5C5	Handel returncode af in parmlist.
B38A-	A9 00	LDA	#\$00	
B38C-	85 48	STA	\$48	IOB set en clear monitor-status.
B38E-	20 7E AE	JSR	\$AE7E	File manager werkruimte in file buffer.
B391-	28	PLP		
B392-	AE 9B B3	LDX	\$B39B	Herstel processor status en stack
B395-	9A	TXS		
B396-	60	RTS		-----
B397-	11 0F	ORA	(\$0F),Y	File manager "kladblaadje"
B399-	00	BRK		
B39A-	00	BRK		
B39B-	E6 00	INC	\$00	S register en directory index
B39D-	01 00	ORA	(\$00,X)	Catalog teller/lock-flag
B39F-	00	BRK		Toewijzingsflag
B3A0-	00	BRK		Track free mask gebruikt door Init
B3A1-	00	BRK		,
B3A2-	FF	???		,
B3A3-	FF	???		Track free mask gebruikt door Init
B3A4-	01 0A	ORA	(\$0A,X)	Decimale conversie tabel
B3A6-	64	???		01, 10, 100
B3A7-	D4	???		Filetabel van catalog ASC "T"
B3A8-	C9 C1	CMP	#\$C1	"IA"
B3AA-	C2	???		"B"
B3AB-	D3	???		"S"
B3AC-	D2	???		"R"
B3AD-	C1 C2	CMP	(\$C2,X)	"AB"
B3AF-	A0 C5	LDY	#\$C5	"EMULOV KSID"
B3B1-	CD D5 CC	CMP	#\$C5	
B3B4-	CF	???		
B3B5-	D6 A0	DEC	\$A0,X	Zie ASCII-tabel. U kunt deze codes zelf wijzigen.
B3B7-	CB	???		
B3B8-	D3	???		
B3B9-	C9 C4	CMP	#\$C4	
B3BB-	04	???		VTOC sector buffer ****
B3BC-	11 0F	ORA	(\$0F),Y	T/S eerste dir sector
B3BE-	03	???		DOS release/versienummer (1,2,3)
B3BF-	00	BRK		
B3C0-	00	BRK		
B3C1-	FE 00 00	INC	\$0000,X	Volumenummer diskette
B3C4-	00	BRK		
B3C5-	00	BRK		
B3C6-	00	BRK		
B3C7-	00	BRK		
B3C8-	00	BRK		
B3C9-	00	BRK		
B3CA-	00	BRK		
B3CB-	00	BRK		
B3CC-	00	BRK		
B3CD-	00	BRK		
B3CE-	00	BRK		
B3CF-	00	BRK		
B3D0-	00	BRK		
B3D1-	00	BRK		
B3D2-	00	BRK		
B3D3-	00	BRK		
B3D4-	00	BRK		

VT0C sector buffer vervolg

B3D5-	00	BRK	
B3D6-	00	BRK	
B3D7-	00	BRK	
B3D8-	00	BRK	
B3D9-	00	BRK	
B3DA-	00	BRK	
B3DB-	00	BRK	
B3DC-	00	BRK	
B3DD-	00	BRK	
B3DE-	00	BRK	
B3DF-	00	BRK	
B3E0-	00	BRK	
B3E1-	00	BRK	
B3E2-	7A	???	
B3E3-	00	BRK	
B3E4-	00	BRK	
B3E5-	00	BRK	
B3E6-	00	BRK	
B3E7-	00	BRK	
B3E8-	00	BRK	
B3E9-	00	BRK	
B3EA-	00	BRK	
B3EB-	0D FF 00	ORA	\$00FF Track welke straks is toe te wijzen
B3EE-	00	BRK	B3EC: + of - looprichting arm.
B3EF-†	23	???	Aantal tracks per disk
B3F0-	10 00	BPL	\$B3F2 Aantal sectoren per track
B3F2-	01 00	ORA	(%00,x) Sectoromvang 2b (100)
B3F4-	00	BRK	Track 0 bit map
B3F5-	00	BRK	
B3F6-	00	BRK	
B3F7-	00	BRK	Track 1 bit map
B3F8-	00	BRK	
B3F9-	00	BRK	
B3FA-	00	BRK	
B3FB-	00	BRK	
B3FC-	00	BRK	
B3FD-	00	BRK	
B3FE-	00	BRK	
B3FF-	00	BRK	
B400-	00	BRK	
B401-	00	BRK	
B402-	00	BRK	† Kan aangepast worden b.v. \$24, mits
B403-	00	BRK	eerder vermelde pointers worden
B404-	00	BRK	afgestemd. (z.a.)
B405-	00	BRK	
B406-	00	BRK	
B407-	00	BRK	
B408-	00	BRK	
B409-	00	BRK	
B40A-	00	BRK	
B40B-	00	BRK	
B40C-	00	BRK	
B40D-	00	BRK	
B40E-	00	BRK	
B40F-	00	BRK	
B410-	00	BRK	
B411-	00	BRK	
B412-	00	BRK	
B413-	00	BRK	
B414-	00	BRK	
B415-	00	BRK	
B416-	00	BRK	→ B47B Track bit map 34

B495-	00	BRK	
B496-	00	BRK	
B497-	00	BRK	
B498-	00	BRK	
B499-	00	BRK	
B49A-	00	BRK	
B49B-	00	BRK	
B49C-	00	BRK	
B49D-	00	BRK	
B49E-	00	BRK	
B49F-	00	BRK	
B4A0-	00	BRK	
B4A1-	00	BRK	
B4A2-	00	BRK	
B4A3-	00	BRK	
B4A4-	00	BRK	
B4A5-	00	BRK	
B4A6-	00	BRK	
B4A7-	00	BRK	
B4A8-	00	BRK	
B4A9-	00	BRK	
B4AA-	00	BRK	
B4AB-	00	BRK	
B4AC-	00	BRK	
B4AD-	00	BRK	
B4AE-	00	BRK	
B4AF-	00	BRK	
B4B0-	00	BRK	
B4B1-	00	BRK	
B4B2-	00	BRK	
B4B3-	00	BRK	
B4B4-	00	BRK	
B4B5-	00	BRK	
B4B6-	00	BRK	
B4B7-	00	BRK	
B4B8-	00	BRK	
B4B9-	00	BRK	
B4BA-	00	BRK	
B4BB-	00	BRK	
B4BC-	11 0E	ORA (\$0E),Y	Directory sector buffer Track/sector volgende dir sector
B4BE-	00	BRK	
B4BF-	00	BRK	
B4C0-	00	BRK	
B4C1-	00	BRK	
B4C2-	00	BRK	
B4C3-	00	BRK	
B4C4-	00	BRK	
B4C5-	00	BRK	
B4C6-	13	???	Eerste Catalog entry en daarvan
B4C7-	0F	???	de T/S List op T13 S f
B4C8-	B2	???	Filetype en lockbit (Applesoft*)
B4C9-	C8	INY	"HELLO" in ASCII
B4CA-	C5 CC	CMP \$CC	
B4CC-	CC CF A0	CPY \$A0CF	
B4CF-	A0 A0	LDY #\$A0	
B4D1-	A0 A0	LDY #\$A0	
B4D3-	A0 A0	LDY #\$A0	
B4D5-	A0 A0	LDY #\$A0	
B4D7-	A0 A0	LDY #\$A0	
B4D9-	A0 A0	LDY #\$A0	
B4DB-	A0 A0	LDY #\$A0	
B4DD-	A0 A0	LDY #\$A0	=spaties -tot B5BA

B5C1-	00	BRK		Vanaf \$B5BB file manager parmlist*
B5C2-	00	BRK		
B5C3-	00	BRK		
B5C4-	00	BRK		
B5C5-	00	BRK		-hier komt b.v. de returncode
B5C6-	00	BRK		
B5C7-	00	BRK		-hier komt b.v. adres f.m. werkruimte
B5C8-	00	BRK		
B5C9-	00	BRK		-hier komt b.v. adres TSL buffer
B5CA-	00	BRK		
B5CB-	00	BRK		-hier komt b.v. adres datasector
B5CC-	00	BRK		buffer
B5CD-	00	BRK		-hier komt b.v. adres volgende
B5CE-	00	BRK		DOS buffer
B5CF-	00	BRK		
B5D0-	00	BRK		File manager werkruimte ***
B5D1-	13	???		eerste TSL track
B5D2-	0F	???		en sector
B5D3-	13	???		huidige TSL t/s
B5D4-	0F	???		
B5D5-	00	BRK		flags 80, 40, 20, 02, 00 write
B5D6-	13	???		Data sector/track
B5D7-	0A	ASL		
B5D8-	01 00	ORA	(\$00,X)	Directory sector index
B5DA-	7A	???		Aantal sectoren in een TSL
B5DB-	00	BRK		
B5DC-	00	BRK		Relatieve sectornummer FSIL
B5DD-	00	BRK		
B5DE-	7A	???		Relatieve sectornummer LastSector
B5DF-	00	BRK		InList.
B5E0-	04	???		Relatieve sectornummer LastRead
B5E1-	00	BRK		
B5E2-	00	BRK		Sectorlengte in bytes hi-lo swap
B5E3-	01 04	ORA	(\$04,X)	Sectorlengte in bytes hi-lo swap
B5E5-	00	BRK		3 bytes sector offset E4,E5,E6
B5E6-	73	???		
B5E7-	00	BRK		
B5E8-	01 00	ORA	(\$00,X)	Recordlengte t.b.v. OPEN
B5EA-	73	???		Recordnummer
B5EB-	04	???		
B5EC-	00	BRK		Byte offset
B5ED-	00	BRK		
B5EE-	06 00	ASL	\$00	Aantal sectoren in file
B5F0-	00	BRK		Sector toewijzingsgebied 6b
B5F1-	00	BRK		
B5F2-	00	BRK		
B5F3-	00	BRK		
B5F4-	00	BRK		
B5F5-	00	BRK		
B5F6-	82	???		Filetype (Applesoft locked)
B5F7-	60	RTS		Slotnummer *16
B5F8-	01 FF	ORA	(\$FF,X)	Drive/Volumenummer compl.
B5FA-	11 00	ORA	(\$00),Y	Tracknummer
B5FC-	00	BRK		vrij
B5FD-	00	BRK		vrij Hier kunt U iets kwijt.
B5FE-	00	BRK		vrij
B5FF-	00	BRK		vrij
B600-	01 A5	ORA	(\$A5,X)	START VAN BOOTFASE 2 RWTS
B602-	27	???		B600 is Boot 1 image, dat bij het
B603-	C9 09	CMP	##09	initialiseren op track 0, sector 0
B605-	D0 18	BNE	\$B61F	komt.
B607-	A5 2B	LDA	\$2B	

B609-	4A	LSR	
B60A-	4A	LSR	
B60B-	4A	LSR	
B60C-	4A	LSR	
B60D-	09 C0	DRA	##C0
B60F-	85 3F	STA	\$3F
B611-	A9 5C	LDA	##5C
B613-	85 3E	STA	\$3E
B615-	18	CLC	
B616-	AD FE 08	LDA	\$08FE
B619-	6D FF 08	ADC	\$08FF
B61C-	8D FE 08	STA	\$08FE
B61F-	AE FF 08	LDX	\$08FF
B622-	30 15	BMI	\$B639
B624-	BD 4D 08	LDA	\$084D,X
B627-	85 3D	STA	\$3D
B629-	CE FF 08	DEC	\$08FF
B62C-	AD FE 08	LDA	\$08FE
B62F-	85 27	STA	\$27
B631-	CE FE 08	DEC	\$08FE
B634-	A6 2B	LDX	\$2B
B636-	6C 3E 00	JMP	(\$003E)
B639-	EE FE 08	INC	\$08FE
B63C-	EE FE 08	INC	\$08FE
B63F-	20 89 FE	JSR	\$FE89
B642-	20 93 FE	JSR	\$FE93
B645-	20 2F FB	JSR	\$FB2F
B648-	A6 2B	LDX	\$2B
B64A-	6C FD 08	JMP	(\$08FD)
B64D-	00	BRK	
B64E-	0D 0B 09	DRA	\$090B
B651-	07	???	
B652-	05 03	DRA	\$03
B654-	01 0E	DRA	(\$0E,X)
B656-	0C	???	
B657-	0A	ASL	
B658-	08	PHP	
B659-	06 04	ASL	\$04
B65B-	02	???	
B65C-	0F	???	
B65D-	00	BRK	
B65E-	20 64 A7	JSR	\$A764
B661-	B0 08	BCS	\$B66B
B663-	A9 00	LDA	##00
B665-	A8	TAY	
B666-	8D 5D B6	STA	\$B65D
B669-	91 40	STA	(\$40),Y
B66B-	AD C5 B5	LDA	\$B5C5
B66E-	4C D2 A6	JMP	\$A6D2
B671-	AD 5D B6	LDA	\$B65D
B674-	F0 08	BEQ	\$B67E
B676-	EE BD B5	INC	\$B5BD
B679-	D0 03	BNE	\$B67E
B67B-	EE BE B5	INC	\$B5BE
B67E-	A9 00	LDA	##00
B680-	8D 5D B6	STA	\$B65D
B683-	4C 46 A5	JMP	\$A546
B686-	8D BC B5	STA	\$B5BC
B689-	20 A8 A6	JSR	\$A6A8
B68C-	20 EA A2	JSR	\$A2EA
B68F-	4C 7D A2	JMP	\$A27D
B692-	A0 13	LDY	##13

RWTS image voor T 0, S 0

 Bootslot
 Set keyboard
 Set video
 Set monitor
 X bootslot
 Bij sommige "beveiligde" diskettes
 vind je hier afwijkingen.

APPEND patch gebied.
 Hier kom je vanuit de Append
 afhandeling.
 Foutafhandeling en juist sluiten
 van de file, indien fout niet
 "end of data" is.

VERIFY patch gebied.
 Verify na Save of Bsave.
 Impliciet uitvoering van Verify

APPEND patch na "End of Data"

B694-	B1 42	LDA	(\$42),Y	
B696-	D0 14	BNE	\$B6AC	
B698-	CB	INY		
B699-	C0 17	CPY	##17	Test de file positie Ø
B69B-	D0 F7	BNE	\$B694	Indien niet Ø dan Append-flag
B69D-	A0 19	LDY	##19	setten.
B69F-	B1 42	LDA	(\$42),Y	
B6A1-	99 A4 B5	STA	\$B5A4,Y	
B6A4-	CB	INY		
B6A5-	C0 1D	CPY	##1D	
B6A7-	D0 F6	BNE	\$B69F	Indien aan begin van file, dan
B6A9-	4C BC A6	JMP	\$A6BC	f.m. parmlist bijwerken.
B6AC-	A2 FF	LDX	##FF	
B6AE-	8E 5D B6	STX	\$B65D	
B6B1-	D0 F6	BNE	\$B6A9	
B6B3-	00	BRK		
B6B4-	00	BRK		
B6B5-	00	BRK		
B6B6-	00	BRK		
B6B7-	00	BRK		
B6B8-	00	BRK		
B6B9-	00	BRK		
B6BA-	00	BRK		
B6BB-	00	BRK		
B6BC-	00	BRK		
B6BD-	00	BRK		
B6BE-	00	BRK		
B6BF-	00	BRK		
B6C0-	00	BRK		
B6C1-	00	BRK		
B6C2-	00	BRK		
B6C3-	00	BRK		
B6C4-	00	BRK		
B6C5-	00	BRK		
B6C6-	00	BRK		
B6C7-	00	BRK		
B6C8-	00	BRK		
B6C9-	00	BRK		
B6CA-	00	BRK		
B6CB-	00	BRK		
B6CC-	00	BRK		
B6CD-	00	BRK		
B6CE-	00	BRK		
B6CF-	00	BRK		
B6D0-	20 5B FC	JSR	\$FC5B	
B6D3-	A9 C2	LDA	##C2	
B6D5-	20 ED FD	JSR	\$FDED	
B6D8-	A9 01	LDA	##01	
B6DA-	20 DA FD	JSR	\$FDDA	
B6DD-	A9 AD	LDA	##AD	
B6DF-	20 ED FD	JSR	\$FDED	
B6E2-	A9 00	LDA	##00	
B6E4-	20 DA FD	JSR	\$FDDA	
B6E7-	60	RTS		-----
B6E8-	00	BRK		
B6E9-	00	BRK		
B6EA-	00	BRK		
B6EB-	00	BRK		
B6EC-	00	BRK		
B6ED-	00	BRK		
B6EE-	00	BRK		
B6EF-	00	BRK		

B6F0-	00	BRK	
B6F1-	00	BRK	
B6F2-	00	BRK	
B6F3-	00	BRK	
B6F4-	00	BRK	
B6F5-	00	BRK	
B6F6-	00	BRK	
B6F7-	00	BRK	
B6F8-	00	BRK	
B6F9-	00	BRK	
B6FA-	00	BRK	
B6FB-	00	BRK	
B6FC-	00	BRK	
B6FD-	00	BRK	
B6FE-	36 09	RDL	\$09,X
B700-	8E E9 B7	STX	\$B7E9
B703-	8E F7 B7	STX	\$B7F7
B706-	A9 01	LDA	#\$01
B708-	8D F8 B7	STA	\$B7F8
B70B-	8D EA B7	STA	\$B7EA
B70E-	AD E0 B7	LDA	\$B7E0
B711-	8D E1 B7	STA	\$B7E1
B714-	A9 02	LDA	#\$02
B716-	8D EC B7	STA	\$B7EC
B719-	A9 04	LDA	#\$04
B71B-	8D ED B7	STA	\$B7ED
B71E-	AC E7 B7	LDY	\$B7E7
B721-	88	DEY	
B722-	8C F1 B7	STY	\$B7F1
B725-	A9 01	LDA	#\$01
B727-	8D F4 B7	STA	\$B7F4
B72A-	8A	TXA	
B72B-	4A	LSR	
B72C-	4A	LSR	
B72D-	4A	LSR	
B72E-	4A	LSR	
B72F-	AA	TAX	
B730-	A9 00	LDA	#\$00
B732-	9D F8 04	STA	\$04F8,X
B735-	9D 78 04	STA	\$0478,X
B738-	20 93 B7	JSR	\$B793
B73B-	A2 FF	LDX	#\$FF
B73D-	9A	TXS	
B73E-	8E EB B7	STX	\$B7EB
B741-	4C CB BF	JMP	\$BFCB
B744-	20 89 FE	JSR	\$FE89
B747-	4C 84 9D	JMP	\$9D84
B74A-	AD E7 B7	LDA	\$B7E7
B74D-	38	SEC	
B74E-	ED F1 B7	SBC	\$B7F1
B751-	8D E1 B7	STA	\$B7E1
B754-	AD E7 B7	LDA	\$B7E7
B757-	8D F1 B7	STA	\$B7F1
B75A-	CE F1 B7	DEC	\$B7F1
B75D-	A9 02	LDA	#\$02
B75F-	8D EC B7	STA	\$B7EC
B762-	A9 04	LDA	#\$04
B764-	8D ED B7	STA	\$B7ED
B767-	A9 02	LDA	#\$02
B769-	8D F4 B7	STA	\$B7F4
B76C-	20 93 B7	JSR	\$B793
B76F-	AD E7 B7	LDA	\$B7E7

B00T2 Fase ***
 Page adres 1ste pagina boot2
 Aantal sectoren in boot2
 B700: Stel slotnummer in
 Drive

T nummer
 S nummer

Commando code
 Creëer nieuwe stack

Daar is-ie
 Read/Write module

Volume
 Set keyboard
 Doe een Coldstart
 Schrijf DOS op track 0-2

Set RWTS parmlist voor het
 wegschrijven van het DOS

Read-Write pages module

B772-	8D FE B6	STA	\$B6FE	
B775-	18	CLC		
B776-	69 09	ADC	#\$09	
B778-	8D F1 B7	STA	\$B7F1	
B77B-	A9 0A	LDA	#\$0A	
B77D-	8D E1 B7	STA	\$B7E1	
B780-	38	SEC		
B781-	E9 01	SBC	#\$01	
B783-	8D FF B6	STA	\$B6FF	
B786-	8D ED B7	STA	\$B7ED	
B789-	20 93 B7	JSR	\$B793	
B78C-	60	RTS		-----
B78D-	00	BRK		vrij
B78E-	00	BRK		vrij
B78F-	00	BRK		vrij
B790-	00	BRK		vrij
B791-	00	BRK		vrij
B792-	00	BRK		vrij
B793-	AD E5 B7	LDA	\$B7E5	Read/Write pages module ***
B796-	AC E4 B7	LDY	\$B7E4	
B799-	20 B5 B7	JSR	\$B7B5	Call RWTS middels extern entry
B79C-	AC ED B7	LDY	\$B7ED	punt.
B79F-	88	DEY		
B7A0-	10 07	BPL	\$B7A9	
B7A2-	A0 0F	LDY	#\$0F	
B7A4-	EA	NOP		
B7A5-	EA	NOP		
B7A6-	CE EC B7	DEC	\$B7EC	
B7A9-	BC ED B7	STY	\$B7ED	
B7AC-	CE F1 B7	DEC	\$B7F1	
B7AF-	CE E1 B7	DEC	\$B7E1	
B7B2-	D0 DF	BNE	\$B793	
B7B4-	60	RTS		-----
B7B5-	08	PHP		Doe interrupts teniet
B7B6-	78	SEI		
B7B7-	20 00 BD	JSR	\$BD00	en CALL RWTS
B7BA-	B0 03	BCS	\$B7BF	
B7BC-	28	PLP		
B7BD-	18	CLC		
B7BE-	60	RTS		-----
B7BF-	28	PLP		
B7C0-	38	SEC		
B7C1-	60	RTS		-----
B7C2-	AD BC B5	LDA	\$B5BC	Set RWTS parmlist voor
B7C5-	8D F1 B7	STA	\$B7F1	het wegschrijven van het DOS
B7C8-	A9 00	LDA	#\$00	
B7CA-	8D F0 B7	STA	\$B7F0	
B7CD-	AD F9 B5	LDA	\$B5F9	
B7D0-	49 FF	EOR	#\$FF	
B7D2-	8D EB B7	STA	\$B7EB	
B7D5-	60	RTS		-----
B7D6-	A9 00	LDA	#\$00	Maak huidige buffer leeg.
B7D8-	AB	TAY		
B7D9-	91 42	STA	(\$42),Y	1 pagina leegmaken en pointers
B7DB-	CB	INY		bijwerken
B7DC-	D0 FB	BNE	\$B7D9	
B7DE-	60	RTS		-----
B7DF-	00	BRK		vrij
B7E0-	1B	???		aantal
B7E1-	00	BRK		pages in boot2 load
B7E2-	0A	ASL		aantal
B7E3-	1B	???		sectoren voor R/W operatie
				aantal
				pages in boot1 load

B7E4-	EB	INX		Pointer naar RWTs parmlist 2b
B7E5-	B7	???		
B7E6-	00	BRK		Pointer naar boot fase 1 2b
B7E7-	B6 01	LDX	\$01,Y	B7E8: RWTs PARMLIST*****
B7E9-	60	RTS		Slot x16
B7EA-	01 FF	ORA	(\$FF,X)	Drive/volume (Ø dan elke V goed)
B7EC-	13	???		Tracknummer
B7ED-	0A	ASL		Sectornummer
B7EE-	FB	???		Pointer naar Device Characteristic
B7EF-	B7	???		Table: DCT
B7F0-	00	BRK		Pointer naar data buffer
B7F1-	96 00	STX	\$00,Y	t.b.v. Read/Write
B7F3-	01 01	ORA	(\$01,X)	Byte teller/Commandocode 1=read
B7F5-	00	BRK		Foutcode
B7F6-	FE 60 01	INC	\$0160,X	Gevonden volumenummer/slot/drive
B7F9-	00	BRK		vrij
B7FA-	00	BRK		vrij
B7FB-	00	BRK		DCT device type moet 01 zijn
B7FC-	01 EF	ORA	(\$EF,X)	Fasen per track (01)/Motor on
B7FE-	DB	CLD		time teller 2b
B7FF-	00	BRK		vrij
B800-	A2 00	LDX	#\$00	PRENIBBLE ROUTINE voor data transfer
B802-	A0 02	LDY	#\$02	Vertaalt 256 x 8 bits bytes in
B804-	88	DEY		342 x 6 bits "bytes"
B805-	B1 3E	LDA	(\$3E),Y	Pointer naar te verwerken page
B807-	4A	LSR		
B808-	3E 00 BC	ROL	\$BC00,X	
B80B-	4A	LSR		
B80C-	3E 00 BC	ROL	\$BC00,X	
B80F-	99 00 BB	STA	\$BB00,Y	
B812-	EB	INX		
B813-	E0 56	CPX	#\$56	
B815-	90 ED	BCC	\$B804	
B817-	A2 00	LDX	#\$00	
B819-	98	TYA		
B81A-	D0 EB	BNE	\$B804	
B81C-	A2 55	LDX	#\$55	
B81E-	BD 00 BC	LDA	\$BC00,X	
B821-	29 3F	AND	#\$3F	
B823-	9D 00 BC	STA	\$BC00,X	
B826-	CA	DEX		
B827-	10 F5	BPL	\$B81E	
B829-	60	RTS		Formaat 00XXXXXX
B82A-	38	SEC		WRITE ROUTINE t.b.v. primaire
B82B-	86 27	STX	\$27	en secundaire buffers.
B82D-	8E 7B 06	STX	\$067B	
B830-	BD 8D C0	LDA	\$C08D,X	
B833-	BD 8E C0	LDA	\$C08E,X	
B836-	30 7C	BMI	\$BBB4	
B838-	AD 00 BC	LDA	\$BC00	
B83B-	85 26	STA	\$26	
B83D-	A9 FF	LDA	#\$FF	
B83F-	9D 8F C0	STA	\$C08F,X	
B842-	1D 8C C0	ORA	\$C08C,X	
B845-	48	PHA		
B846-	68	PLA		
B847-	EA	NOP		
B848-	A0 04	LDY	#\$04	
B84A-	48	PHA		
B84B-	68	PLA		
B84C-	20 B9 BB	JSR	\$BBB9	
B84F-	88	DEY		

B850-	D0 F8	BNE	\$B84A	
B852-	A9 D5	LDA	#\$D5	Schrijf 5 autosync bytes, start data
B854-	20 B8 B8	JSR	\$B8B8	bytes (\$D5, \$AA, \$AD), 342 data bytes
B857-	A9 AA	LDA	#\$AA	een 1 byte checksum, en
B859-	20 B8 B8	JSR	\$B8B8	eindmarkeringen (\$DE, \$AA, \$EB)
B85C-	A9 AD	LDA	#\$AD	
B85E-	20 B8 B8	JSR	\$B8B8	
B861-	98	TYA		
B862-	A0 56	LDY	#\$56	
B864-	D0 03	BNE	\$B869	
B866-	B9 00 BC	LDA	\$BC00,Y	
B869-	59 FF BB	EOR	\$B8FF,Y	
B86C-	AA	TAX		
B86D-	BD 29 BA	LDA	\$BA29,X	Write Translate Tabel
B870-	A6 27	LDX	\$27	
B872-	9D 8D C0	STA	\$C08D,X	
B875-	BD 8C C0	LDA	\$C08C,X	
B878-	88	DEY		
B879-	D0 EB	BNE	\$B866	
B87B-	A5 26	LDA	\$26	
B87D-	EA	NOP		
B87E-	59 00 BB	EOR	\$BB00,Y	
B881-	AA	TAX		
B882-	BD 29 BA	LDA	\$BA29,X	
B885-	AE 78 06	LDX	\$0678	
B888-	9D 8D C0	STA	\$C08D,X	
B88B-	BD 8C C0	LDA	\$C08C,X	
B88E-	B9 00 BB	LDA	\$BB00,Y	
B891-	C8	INY		
B892-	D0 EA	BNE	\$B87E	
B894-	AA	TAX		
B895-	BD 29 BA	LDA	\$BA29,X	
B898-	A6 27	LDX	\$27	
B89A-	20 B8 B8	JSR	\$B8B8	
B89D-	A9 DE	LDA	#\$DE	
B89F-	20 B8 B8	JSR	\$B8B8	
B8A2-	A9 AA	LDA	#\$AA	
B8A4-	20 B8 B8	JSR	\$B8B8	
B8A7-	A9 EB	LDA	#\$EB	
B8A9-	20 B8 B8	JSR	\$B8B8	
B8AC-	A9 FF	LDA	#\$FF	
B8AE-	20 B8 B8	JSR	\$B8B8	
B8B1-	BD 8E C0	LDA	\$C08E,X	
B8B4-	BD 8C C0	LDA	\$C08C,X	
B8B7-	60	RTS		-----
B8B8-	18	CLC		Write byte subroutine ****
B8B9-	48	PHA		
B8BA-	68	PLA		
B8BB-	9D 8D C0	STA	\$C08D,X	Timing code wordt benut voor
B8BE-	1D 8C C0	ORA	\$C08C,X	het wegschrijven. 32 cycles
B8C1-	60	RTS		interval.
B8C2-	A0 00	LDY	#\$00	-----
B8C4-	A2 56	LDX	#\$56	POSTNIBBLE Subroutine
B8C6-	CA	DEX		Vertaalt 342 x 6 bits "bytes"
B8C7-	30 FB	BMI	\$B8C4	in 256 x 8 bits bytes.
B8C9-	B9 00 BB	LDA	\$BB00,Y	
B8CC-	5E 00 BC	LSR	\$BC00,X	
B8CF-	2A	ROL		
B8D0-	5E 00 BC	LSR	\$BC00,X	
B8D3-	2A	ROL		
B8D4-	91 3E	STA	(\$3E),Y	
B8D6-	C8	INY		

B8D7-	C4 26	CPY	\$26	
B8D9-	D0 EB	BNE	\$B8C6	
B8DB-	60	RTS		-----
B8DC-	A0 20	LDY	#\$20	READ subroutine ***
B8DE-	88	DEY		Lees sector met data van disk en
B8DF-	F0 61	BEQ	\$B942	gebruikt de buffers.
B8E1-	BD 8C C0	LDA	\$C08C,X	
B8E4-	10 FB	BPL	\$B8E1	
B8E6-	49 D5	EOR	#\$D5	
B8E8-	D0 F4	BNE	\$B8DE	
B8EA-	EA	NOP		
B8EB-	BD 8C C0	LDA	\$C08C,X	
B8EE-	10 FB	BPL	\$B8EB	
B8F0-	C9 AA	CMP	#\$AA	
B8F2-	D0 F2	BNE	\$B8E6	
B8F4-	A0 56	LDY	#\$56	
B8F6-	BD 8C C0	LDA	\$C08C,X	
B8F9-	10 FB	BPL	\$B8F6	
B8FB-	C9 AD	CMP	#\$AD	
B8FD-	D0 E7	BNE	\$B8E6	
B8FF-	A9 00	LDA	#\$00	
B901-	88	DEY		
B902-	84 26	STY	\$26	
B904-	BC 8C C0	LDY	\$C08C,X	
B907-	10 FB	BPL	\$B904	
B909-	59 00 BA	EOR	\$BA00,Y	
B90C-	A4 26	LDY	\$26	
B90E-	99 00 BC	STA	\$BC00,Y	
B911-	D0 EE	BNE	\$B901	
B913-	84 26	STY	\$26	
B915-	BC 8C C0	LDY	\$C08C,X	
B918-	10 FB	BPL	\$B915	
B91A-	59 00 BA	EOR	\$BA00,Y	
B91D-	A4 26	LDY	\$26	
B91F-	99 00 BB	STA	\$BB00,Y	
B922-	C8	INY		"Beveiligde" diskettes kunnen
B923-	D0 EE	BNE	\$B913	hierop heel lelijk doen, als
B925-	BC 8C C0	LDY	\$C08C,X	we ze met het standaard DOS
B928-	10 FB	BPL	\$B925	willen gebruiken.
B92A-	D9 00 BA	CMP	\$BA00,Y	
B92D-	D0 13	BNE	\$B942	
B92F-	BD 8C C0	LDA	\$C08C,X	
B932-	10 FB	BPL	\$B92F	
B934-	C9 DE	CMP	#\$DE	
B936-	D0 0A	BNE	\$B942	
B938-	EA	NOP		
B939-	BD 8C C0	LDA	\$C08C,X	
B93C-	10 FB	BPL	\$B939	
B93E-→	C9 AA	CMP	#\$AA	→A9 00 is mogelijk en kan
B940-	F0 5C	BEQ	\$B99E	ertoe leiden, dat "beveiliging"
B942-	38	SEC		het begeeft.
B943-	60	RTS		
B944-	A0 FC	LDY	#\$FC	
B946-	84 26	STY	\$26	
B948-	C8	INY		
B949-	D0 04	BNE	\$B94F	
B94B-	E6 26	INC	\$26	
B94D-	F0 F3	BEQ	\$B942	
B94F-	BD 8C C0	LDA	\$C08C,X	
B952-	10 FB	BPL	\$B94F	
B954-	C9 D5	CMP	#\$D5	
B956-	D0 F0	BNE	\$B948	

B958-	EA	NOP	
B959-	BD 8C C0	LDA	\$C08C,X
B95C-	10 FB	BPL	\$B959
B95E-	C9 AA	CMP	##AA
B960-	D0 F2	BNE	\$B954
B962-	A0 03	LDY	##03
B964-	BD 8C C0	LDA	\$C08C,X
B967-	10 FB	BPL	\$B964
B969-	C9 96	CMP	##96
B96B-	D0 E7	BNE	\$B954
B96D-	A9 00	LDA	##00
B96F-	85 27	STA	\$27
B971-	BD 8C C0	LDA	\$C08C,X
B974-	10 FB	BPL	\$B971
B976-	2A	ROL	
B977-	85 26	STA	\$26
B979-	BD 8C C0	LDA	\$C08C,X
B97C-	10 FB	BPL	\$B979
B97E-	25 26	AND	\$26
B980-	99 2C 00	STA	\$002C,Y
B983-	45 27	EOR	\$27
B985-	88	DEY	
B986-	10 E7	BPL	\$B96F
B988-	AB	TAY	
B989-	D0 B7	BNE	\$B942
B98B-	BD 8C C0	LDA	\$C08C,X
B98E-	10 FB	BPL	\$B98B
B990-	C9 DE	CMP	##DE
B992-	D0 AE	BNE	\$B942
B994-	EA	NOP	
B995-	BD 8C C0	LDA	\$C08C,X
B998-	10 FB	BPL	\$B995
B99A-→	C9 AA	CMP	##AA
B99C-	D0 A4	BNE	\$B942
B99E-	18	CLC	
B99F-	60	RTS	
B9A0-	86 2B	STX	\$2B
B9A2-	85 2A	STA	\$2A
B9A4-	CD 78 04	CMP	\$047B
B9A7-	F0 53	BEQ	\$B9FC
B9A9-	A9 00	LDA	##00
B9AB-	85 26	STA	\$26
B9AD-	AD 78 04	LDA	\$047B
B9B0-	85 27	STA	\$27
B9B2-	38	SEC	
B9B3-	E5 2A	SBC	\$2A
B9B5-	F0 33	BEQ	\$B9EA
B9B7-	B0 07	BCS	\$B9C0
B9B9-	49 FF	EOR	##FF
B9BB-	EE 78 04	INC	\$047B
B9BE-	90 05	BCC	\$B9C5
B9C0-	69 FE	ADC	##FE
B9C2-	CE 78 04	DEC	\$047B
B9C5-	C5 26	CMP	\$26
B9C7-	90 02	BCC	\$B9CB
B9C9-	A5 26	LDA	\$26
B9CB-	C9 0C	CMP	##0C
B9CD-	B0 01	BCS	\$B9D0
B9CF-	AB	TAY	
B9D0-	38	SEC	
B9D1-	20 EE B9	JSR	\$B9EE
B9D4-	B9 11 BA	LDA	\$BA11,Y

READ ADRES FIELD Subroutine ***

Lees start adres merktekens
(\$D5,\$AA,\$96), adres informatie
(V, T/S, checksum) en afsluiting
(\$DE,\$AA)

Ook hier geven sommige "beveiligingen" problemen.

→A9:00 kan problemen oplossen.
In samenhang met B93E

ARM BESTURING 'SEEKABS'

Zoek de juiste track.

Acc=track (half voor eenfase drive)

B9D7-	20 00 BA	JSR	\$BA00	
B9DA-	A5 27	LDA	\$27	
B9DC-	18	CLC		
B9DD-	20 F1 B9	JSR	\$B9F1	
B9E0-	B9 1D BA	LDA	\$BA1D,Y	
B9E3-	20 00 BA	JSR	\$BA00	
B9E6-	E6 26	INC	\$26	
B9E8-	D0 C3	BNE	\$B9AD	
B9EA-	20 00 BA	JSR	\$BA00	
B9ED-	18	CLC		
B9EE-	AD 78 04	LDA	\$0478	
B9F1-	29 03	AND	#\$03	
B9F3-	2A	RDL		
B9F4-	05 2B	ORA	\$2B	
B9F6-	AA	TAX		
B9F7-	BD 80 C0	LDA	\$C080,X	
B9FA-	A6 2B	LDX	\$2B	
B9FC-	60	RTS		
B9FD-	AA	TAX		
B9FE-	A0 A0	LDY	#\$A0	
BA00-	A2 11	LDX	#\$11	
BA02-	CA	DEX		
BA03-	D0 FD	BNE	\$BA02	
BA05-	E6 46	INC	\$46	
BA07-	D0 02	BNE	\$BA0B	
BA09-	E6 47	INC	\$47	
BA0B-	38	SEC		
BA0C-	E9 01	SBC	#\$01	
BA0E-	D0 F0	BNE	\$BA00	
BA10-	60	RTS		
BA11-	01 30	ORA	(\$30,X)	
BA13-	28	PLP		
BA14-	24 20	BIT	\$20	
BA16-	1E 1D 1C	ASL	\$1C1D,X	
BA19-	1C	???		
BA1A-	1C	???		
BA1B-	1C	???		
BA1C-	1C	???		
BA1D-	70 2C	BVS	\$BA4B	
BA1F-	26 22	ROL	\$22	
BA21-	1F	???		
BA22-	1E 1D 1C	ASL	\$1C1D,X	
BA25-	1C	???		
BA26-	1C	???		
BA27-	1C	???		
BA28-	1C	???		
BA29-	96 97	STX	\$97,Y	
BA2B-	9A	TXS		
BA2C-	9B	???		
BA2D-	9D 9E 9F	STA	\$9F9E,X	
BA30-	A6 A7	LDX	\$A7	
BA32-	AB	???		
BA33-	AC AD AE	LDY	\$AEAD	
BA36-	AF	???		
BA37-	B2	???		
BA38-	B3	???		
BA39-	B4 B5	LDY	\$B5,X	
BA3B-	B6 B7	LDX	\$B7,Y	
BA3D-	B9 BA BB	LDA	\$BBBA,Y	
BA40-	BC BD BE	LDY	\$BEBD,X	
BA43-	BF	???		
BA44-	CB	???		

Arm Vertragings Routine
Vertraagt een gegeven waarde tot
een veelvoud van 100 μ Sec.

\$46,47 bevatten \$EF,D8 uit DCT

Vertragingstabel armbeweging

Fase 0 en 1 vertragingssets
voor stappenmotor

WRITE Vertalings Routine ***
Bevat 6 bits "bytes", waarmee
8 bits bytes worden vertaald.
Waarden tussen \$96 en \$FF.

BA45-	CD CE CF	CMP	\$CFCE	
BA48-	D3	???		
BA49-	D6 D7	DEC	\$D7, X	
BA4B-	D9 DA DB	CMP	\$DBDA, Y	
BA4E-	DC	???		
BA4F-	DD DE DF	CMP	\$DFDE, X	
BA52-	E5 E6	SBC	\$E6	
BA54-	E7	???		
BA55-	E9 EA	SBC	##EA	
BA57-	EB	???		
BA58-	EC ED EE	CPX	##EED	
BA5B-	EF	???		
BA5C-	F2	???		
BA5D-	F3	???		
BA5E-	F4	???		
BA5F-	F5 F6	SBC	\$F6, X	
BA61-	F7	???		
BA62-	F9 FA FB	SBC	\$FBFA, Y	
BA65-	FC	???		
BA66-	FD FE FF	SBC	\$FFFE, X	
BA69-	B3	???		READ Vertalings Routine ***
BA6A-	B3	???		vrij
BA6B-	A0 E0	LDY	##E0	
BA6D-	B3	???		
BA6E-	C3	???		
BA6F-	C5 B3	CMP	\$B3	
BA71-	A0 E0	LDY	##E0	
BA73-	B3	???		
BA74-	C3	???		
BA75-	C5 B3	CMP	\$B3	
BA77-	A0 E0	LDY	##E0	
BA79-	B3	???		
BA7A-	B3	???		
BA7B-	C5 AA	CMP	\$AA	
BA7D-	A0 B2	LDY	##B2	
BA7F-	B3	???		
BA80-	B3	???		
BA81-	C5 AA	CMP	\$AA	
BA83-	A0 B2	LDY	##B2	
BA85-	C5 B3	CMP	\$B3	
BA87-	B3	???		
BA88-	AA	TAX		
BA89-	BB	DEY		
BABA-	B2	???		
BABB-	C5 B3	CMP	\$B3	
BABD-	B3	???		
BABE-	AA	TAX		
BABF-	BB	DEY		
BA90-	B2	???		
BA91-	C5 C4	CMP	\$C4	
BA93-	B3	???		
BA94-	B0 BB	BCS	\$BA1E	
BA96-	00	BRK		vrij
BA97-	01 9B	ORA	(\$9B, X)	
BA99-	99 02 03	STA	\$0302, Y	
BA9C-	9C	???		
BA9D-	04	???		Nibble vertalingsroutine met
BA9E-	05 06	ORA	\$06	codes tussen \$96 en \$FF
BAA0-	A0 A1	LDY	##A1	
BAA2-	A2 A3	LDX	##A3	
BAA4-	A4 A5	LDY	\$A5	
BAA6-	07	???		

BAA7-	08	PHP	
BAA8-	AB	TAY	
BAA9-	A9 AA	LDA	#\$AA
BAA8-	09 0A	ORA	#\$0A
BAAD-	0B	???	
BAAE-	0C	???	
BAAF-	0D B0 B1	ORA	\$B1B0
BAB2-	0E 0F 10	ASL	\$100F
BAB5-	11 12	ORA	(\$12),Y
BAB7-	13	???	
BAB8-	B8	CLV	
BAB9-	14	???	
BABA-	15 16	ORA	\$16,X
BABC-	17	???	
BABD-	18	CLC	
BABE-	19 1A C0	ORA	\$C01A,Y
BAC1-	C1 C2	CMP	(\$C2,X)
BAC3-	C3	???	
BAC4-	C4 C5	CPY	\$C5
BAC6-	C6 C7	DEC	\$C7
BAC8-	C8	INY	
BAC9-	C9 CA	CMP	#\$CA
BACB-	1B	???	
BACC-	CC 1C 1D	CPY	\$1D1C
BACF-	1E D0 D1	ASL	\$D1D0,X
BAD2-	D2	???	
BAD3-	1F	???	
BAD4-	D4	???	
BAD5-	D5 20	CMP	\$20,X
BAD7-	21 D8	AND	(\$D8,X)
BAD9-	22	???	
BADA-	23	???	
BADB-	24 25	BIT	\$25
BADD-	26 27	ROL	\$27
BADF-	28	PLP	
BAE0-	E0 E1	CPX	#\$E1
BAE2-	E2	???	
BAE3-	E3	???	
BAE4-	E4 29	CPX	\$29
BAE6-	2A	ROL	
BAE7-	2B	???	
BAE8-	E8	INX	
BAE9-	2C 2D 2E	BIT	\$2E2D
BAEC-	2F	???	
BAED-	30 31	BMI	\$\$B20
BAEF-	32	???	
BAF0-	F0 F1	BEQ	\$BAE3
BAF2-	33	???	
BAF3-	34	???	
BAF4-	35 36	AND	\$36,X
BAF6-	37	???	
BAF7-	38	SEC	
BAF8-	F8	SED	
BAF9-	39 3A 3B	AND	\$3B3A,Y
BAFC-	3C	???	
BAFD-	3D 3E 3F	AND	\$3F3E,X
BB00-	11 00	ORA	(\$00),Y
BB02-	01 03	ORA	(\$03,X)
BB04-	03	???	
BB05-	00	BRK	
BB06-	2C 00 05	BIT	\$0500
BB09-	03	???	

Begin Primaire Buffer

Loopt tot \$BBFF door.

BC21-	00	BRK	
BC22-	00	BRK	\$BC00 Start Secundaire Buffer
BC23-	00	BRK	
BC24-	00	BRK	
BC25-	00	BRK	
BC26-	00	BRK	
BC27-	00	BRK	
BC28-	00	BRK	
BC29-	00	BRK	
BC2A-	00	BRK	
BC2B-	00	BRK	
BC2C-	00	BRK	
BC2D-	00	BRK	
BC2E-	00	BRK	
BC2F-	00	BRK	
BC30-	00	BRK	
BC31-	00	BRK	
BC32-	00	BRK	
BC33-	00	BRK	
BC34-	00	BRK	
BC35-	00	BRK	
BC36-	00	BRK	
BC37-	00	BRK	
BC38-	00	BRK	
BC39-	00	BRK	
BC3A-	00	BRK	
BC3B-	00	BRK	
BC3C-	00	BRK	
BC3D-	00	BRK	
BC3E-	00	BRK	
BC3F-	00	BRK	
BC40-	00	BRK	
BC41-	00	BRK	
BC42-	00	BRK	
BC43-	00	BRK	
BC44-	00	BRK	
BC45-	00	BRK	
BC46-	00	BRK	
BC47-	00	BRK	
BC48-	00	BRK	
BC49-	00	BRK	
BC4A-	00	BRK	
BC4B-	00	BRK	
BC4C-	00	BRK	
BC4D-	00	BRK	
BC4E-	00	BRK	
BC4F-	00	BRK	
BC50-	00	BRK	
BC51-	00	BRK	
BC52-	00	BRK	
BC53-	00	BRK	
BC54-	00	BRK	
BC55-	00	BRK	Einde vrije buffer ruimte
BC56-	38	SEC	Schrijf Adresfield tijdens INIT
BC57-	BD 8D C0	LDA	%C08D,X
BC5A-	BD 8E C0	LDA	%C08E,X
BC5D-	30 5E	BMI	%BCBD
BC5F-✓	A9 FF	LDA	##FF
BC61-	9D 8F C0	STA	%C08F,X
BC64-	DD 8C C0	CMP	%C08C,X
BC67-	48	PHA	
BC68-	68	PLA	

Schrijf autosync bytes uit Y-reg

✓\$FE zorgt voor "beveiliging"

BC69-	20 C3 BC	JSR	\$BCC3	
BC6C-	20 C3 BC	JSR	\$BCC3	
BC6F-	9D 8D C0	STA	\$C08D,X	
BC72-	DD 8C C0	CMP	\$C08C,X	
BC75-	EA	NOP		
BC76-	88	DEY		
BC77-	D0 F0	BNE	\$BC69	
BC79-	A9 D5	LDA	#\$D5	Start adres markeringen
BC7B-	20 D5 BC	JSR	\$BCD5	
BC7E-	A9 AA	LDA	##AA	.
BC80-	20 D5 BC	JSR	\$BCD5	
BC83-	A9 96	LDA	##96	.
BC85-	20 D5 BC	JSR	\$BCD5	
BC88-	A5 41	LDA	\$41	Vol nummer
BC8A-	20 C4 BC	JSR	\$BCC4	
BC8D-	A5 44	LDA	\$44	Track nummer
BC8F-	20 C4 BC	JSR	\$BCC4	
BC92-	A5 3F	LDA	\$3F	Sector nummer
BC94-	20 C4 BC	JSR	\$BCC4	
BC97-	A5 41	LDA	\$41	
BC99-	45 44	EOR	\$44	
BC9B-	45 3F	EOR	\$3F	
BC9D-	48	PHA		
BC9E-	4A	LSR		
BC9F-	05 3E	ORA	\$3E	
BCA1-	9D 8D C0	STA	\$C08D,X	
BCA4-	BD 8C C0	LDA	\$C08C,X	
BCA7-	68	PLA		
BCA8-	09 AA	ORA	##AA	
BCAA-	20 D4 BC	JSR	\$BCD4	
BCAD-✓	A9 DE	LDA	##DE	✓\$DF zorgt voor "beveiliging"
BCAF-	20 D5 BC	JSR	\$BCD5	
BCB2-	A9 AA	LDA	##AA	Afsluitmarkering kan voor beveili-
BCB4-	20 D5 BC	JSR	\$BCD5	ging worden gebruikt. INIT met
BCB7-	A9 EB	LDA	##EB	te beveiligen programma in geheugen
BCB9-	20 D5 BC	JSR	\$BCD5	en breng dan de "beveiligingen" aan
BCBC-	18	CLC		waarna INIT kan plaatsvinden.
BCBD-	BD 8E C0	LDA	\$C08E,X	Dus: Laadt programma met standaard
BCC0-	BD 8C C0	LDA	\$C08C,X	DOS, dan wijzigen en INIT.
BCC3-	60	RTS		-----
BCC4-	48	PHA		WRITE dubbele byte subroutine
BCC5-	4A	LSR		Timing code uit 32 cycles
BCC6-	05 3E	ORA	\$3E	interval.
BCC8-	9D 8D C0	STA	\$C08D,X	
BCCB-	DD 8C C0	CMP	\$C08C,X	
BCCE-	68	PLA		
BCCF-	EA	NOP		
BCD0-	EA	NOP		
BCD1-	EA	NOP		
BCD2-	09 AA	ORA	##AA	
BCD4-	EA	NOP		
BCD5-	EA	NOP		
BCD6-	48	PHA		
BCD7-	68	PLA		
BCD8-	9D 8D C0	STA	\$C08D,X	
BCDB-	DD 8C C0	CMP	\$C08C,X	
BCDE-	60	RTS		-----
BCDF-	88	DEY		BCDF-BCFF vrij
BCE0-	A5 E8	LDA	\$E8	Geschikt voor "beveiligingen"
BCE2-	91 A0	STA	(\$A0),Y	Bijv.: \$A4E2: 4C DF BC, dan
BCE4-	94 88	STY	\$88,X	\$BCDF; A9 FF 85 D6 6C 58 9D
BCE6-	96 E8	STX	\$E8,Y	Bekijk hiermee uw Basic programma!

BCEB-	91 A0	STA	(\$A0),Y	
BCEA-	94 88	STY	\$88,X	
BCEC-	96 91	STX	\$91,Y	En: \$AE3A: 4C EC BC dan \$BCEC:
BCEE-	91 C8	STA	(\$C8),Y	20 1B FD C9 9B F0 01 60 4C 2C 6E
BCF0-	94 D0	STY	\$D0,X	Onderbreekt langé Catalog met
BCF2-	96 91	STX	\$91,Y	druk op ESC-toets.
BCF4-	91 C8	STA	(\$C8),Y	
BCF6-	94 D0	STY	\$D0,X	
BCF8-	96 91	STX	\$91,Y	
BCFA-	A3	???		
BCFB-	C8	INY		
BCFC-	A0 A5	LDY	#\$A5	
BCFE-	85 A4	STA	\$A4	
BD00-	84 48	STY	\$48	RWTS HOOFD ROUTINES *****
BD02-	85 49	STA	\$49	Y en A naar \$48,49 pointer IOB
BD04-	A0 02	LDY	#\$02	
BD06-	8C F8 06	STY	\$06F8	
BD09-	A0 04	LDY	#\$04	Zoekparameter, mag niet lager voor
BD0B-	8C F8 04	STY	\$04F8	snellere disk access.
BD0E-	A0 01	LDY	#\$01	
BD10-	B1 48	LDA	(\$48),Y	
BD12-	AA	TAX		
BD13-	A0 0F	LDY	#\$0F	
BD15-	D1 48	CMP	(\$48),Y	Is slot gewijzigd ?
BD17-	F0 1B	BEQ	\$BD34	
BD19-	8A	TXA		Update slot in IOB
BD1A-	48	PHA		
BD1B-	B1 48	LDA	(\$48),Y	
BD1D-	AA	TAX		
BD1E-	68	PLA		
BD1F-	48	PHA		
BD20-	91 48	STA	(\$48),Y	
BD22-	BD 8E C0	LDA	\$C08E,X	Wacht tot oorspronkelijke drive
BD25-	A0 08	LDY	#\$08	stil staat.
BD27-	BD 8C C0	LDA	\$C08C,X	
BD2A-	DD 8C C0	CMP	\$C08C,X	
BD2D-	D0 F6	BNE	\$BD25	
BD2F-	88	DEY		
BD30-	D0 F8	BNE	\$BD2A	
BD32-	68	PLA		
BD33-	AA	TAX		
BD34-	BD 8E C0	LDA	\$C08E,X	
BD37-	BD 8C C0	LDA	\$C08C,X	Set READ mode en controleert
BD3A-	A0 08	LDY	#\$08	of drive draait.
BD3C-	BD 8C C0	LDA	\$C08C,X	
BD3F-	48	PHA		
BD40-	68	PLA		
BD41-	48	PHA		
BD42-	68	PLA		
BD43-	8E F8 05	STX	\$05F8	
BD46-	DD 8C C0	CMP	\$C08C,X	
BD49-	D0 03	BNE	\$BD4E	
BD4B-	88	DEY		
BD4C-	D0 EE	BNE	\$BD3C	
BD4E-	08	PHP		
BD4F-	BD 89 C0	LDA	\$C089,X	
BD52-	A0 06	LDY	#\$06	
BD54-	B1 48	LDA	(\$48),Y	Pointers uit IOB naar zero page
BD56-	99 36 00	STA	\$0036,Y	
BD59-	C8	INY		
BD5A-	C0 0A	CPY	#\$0A	
BD5C-	D0 F6	BNE	\$BD54	

BD5E-	A0 03	LDY	##03	
BD60-	B1 3C	LDA	(\$3C),Y	DCI naar \$3C/D
BD62-	B5 47	STA	\$47	Motor on time \$D8
BD64-	A0 02	LDY	##02	
BD66-	B1 48	LDA	(\$48),Y	
BD68-	A0 10	LDY	##10	Is drive veranderd ?
BD6A-	D1 48	CMP	(\$48),Y	
BD6C-	F0 06	BEQ	\$BD74	Nee.
BD6E-	91 48	STA	(\$48),Y	Anders resultaat omzetten.
BD70-	2B	PLP		
BD71-	A0 00	LDY	##00	
BD73-	0B	PHP		Kies juiste drive.
BD74-	6A	ROR		
BD75-	90 05	BCC	\$BD7C	
BD77-	BD 8A C0	LDA	\$C08A,X	
BD7A-	B0 03	BCS	\$BD7F	
BD7C-	BD 8B C0	LDA	\$C08B,X	1= drive 1, 0= drive 2
BD7F-	66 35	ROR	\$35	Save hem als hi-bit van \$35
BD81-	2B	PLP		
BD82-	0B	PHP		
BD83-	D0 0B	BNE	\$BD90	Drive is aan.
BD85-	A0 07	LDY	##07	
BD87-	20 00 BA	JSR	\$BA00	Wacht anders tot drive condensator ontlaadt.
BD8A-	8B	DEY		
BD8B-	D0 FA	BNE	\$BD87	
BD8D-	AE F8 05	LDX	\$05F8	
BD90-	A0 04	LDY	##04	Naar welke track?
BD92-	B1 48	LDA	(\$48),Y	
BD94-	20 5A BE	JSR	\$BE5A	
BD97-	2B	PLP		
BD98-	D0 11	BNE	\$BDAB	
BD9A-	A4 47	LDY	\$47	
BD9C-	10 0D	BPL	\$BDAB	
BD9E-	A0 12	LDY	##12	
BDA0-	8B	DEY		
BDA1-	D0 FD	BNE	\$BDA0	
BDA3-	E6 46	INC	\$46	
BDA5-	D0 F7	BNE	\$BD9E	
BDA7-	E6 47	INC	\$47	
BDA9-	D0 F3	BNE	\$BD9E	
BDAB-	A0 0C	LDY	##0C	TRYTRK Subroutine
BDAD-	B1 48	LDA	(\$48),Y	Welke commando code vigeert?
BDAF-	F0 5A	BEQ	\$BE0B	
BDB1-	C9 04	CMP	##04	
BDB3-	F0 58	BEQ	\$BE0D	Moet de disk geïnitieerd?
BDB5-	6A	ROR		
BDB6-	0B	PHP		
BDB7-	B0 03	BCS	\$BDBC	
BDB9-	20 00 B8	JSR	\$B800	Voor Write is eerst naar Prenibble.
BDBC-	A0 30	LDY	##30	Controleer het verloop van de handelingen.
BDBE-	8C 78 05	STY	\$0578	
BDC1-	AE F8 05	LDX	\$05F8	
BDC4-	20 44 B9	JSR	\$B944	
BDC7-	90 24	BCC	\$BDED	READ was goed.
BDC9-	CE 78 05	DEC	\$0578	Werk teller bij en probeer opnieuw.
BDCC-	10 F3	BPL	\$BDC1	
BDCE-	AD 78 04	LDA	\$0478	
BDD1-	48	PHA		
BDD2-	A9 60	LDA	##60	
BDD4-	20 95 BE	JSR	\$BE95	
BDD7-	CE F8 06	DEC	\$06F8	
BDDA-	F0 2B	BEQ	\$BE04	Oh, oh drive error!

BDDC-	A9 04	LDA	#\$04	
BDDE-	8D F8 04	STA	\$04F8	
BDE1-	A9 00	LDA	#\$00	
BDE3-	20 5A BE	JSR	\$BE5A	
BDE6-	68	PLA		
BDE7-	20 5A BE	JSR	\$BE5A	
BDEA-	4C BC BD	JMP	\$BDBC	
BDED-	A4 2E	LDY	\$2E	Verifieer juiste track
BDEF-	CC 78 04	CPY	\$0478	
BDF2-	F0 1C	BEQ	\$BE10	OK!
BDF4-	AD 78 04	LDA	\$0478	Anders moeten wij corrigeren.
BDF7-	48	PHA		
BDF8-	98	TYA		
BDF9-	20 95 BE	JSR	\$BE95	
BDFC-	68	PLA		
BDFD-	CE F8 04	DEC	\$04F8	
BE00-	D0 E5	BNE	\$BDE7	
BE02-	F0 CA	BEQ	\$BDCE	
BE04-	68	PLA		
BE05-	A9 40	LDA	\$\$\$40	Maak status en stack reg. schoon en set drive error in Acc \$40
BE07-	28	PLP		
BE08-	4C 48 BE	JMP	\$BE48	Foutje wegwerken!
BE0B-	F0 39	BEQ	\$BE46	
BE0D-	4C AF BE	JMP	\$BEAF	Naar INIT afhandeling
BE10-	A0 03	LDY	#\$03	Controleer gevonden volume
BE12-	B1 48	LDA	(\$48),Y	t.o.v. verwacht volume.
BE14-	48	PHA		\$BE10: 4C 26 BE kan ook, dan geen Vol check.
BE15-	A5 2F	LDA	\$2F	
BE17-	A0 0E	LDY	#\$0E	
BE19-	91 48	STA	(\$48),Y	
BE1B-	68	PLA		
BE1C-	F0 08	BEQ	\$BE26	
BE1E-	C5 2F	CMP	\$2F	
BE20-	F0 04	BEQ	\$BE26	Geen volume, dan geen fout.
BE22-	A9 20	LDA	#\$20	Volume mismatch code ?
BE24-	D0 E1	BNE	\$BE07	Dat was jammer.
BE26-	A0 05	LDY	#\$05	Is dit de juiste sector ?
BE28-	B1 48	LDA	(\$48),Y	
BE2A-	A8	TAY		
BE2B-	B9 B8 BF	LDA	\$BFBB,Y	Naar sector interleaving tabel
BE2E-	C5 2D	CMP	\$2D	
BE30-	D0 97	BNE	\$BDC9	
BE32-	28	PLP		
BE33-	90 1C	BCC	\$BE51	Is de sector juist gekozen, en
BE35-	20 DC B8	JSR	\$B8DC	is de code "schrijven"?
BE38-	08	PHP		Anders moet er data gelezen worden.
BE39-	B0 BE	BCS	\$BDC9	
BE3B-	28	PLP		
BE3C-	A2 00	LDX	#\$00	Is "read" goed verlopen,
BE3E-	86 26	STX	\$26	
BE40-	20 C2 B8	JSR	\$B8C2	Exit via postnibble
BE43-	AE F8 05	LDX	\$05F8	
BE46-	18	CLC		Skip carry
BE47-	24 38	BIT	\$38	in BE48 - sla foutafh. over.
BE49-	A0 0D	LDY	#\$0D	
BE4B-	91 48	STA	(\$48),Y	A heeft returncode
BE4D-	BD B8 C0	LDA	\$C0B8,X	Motor af.
BE50-	60	RTS		-----
BE51-	20 2A B8	JSR	\$B82A	Schrijf een sector ****
BE54-	90 F0	BCC	\$BE46	Gedaan en goed, dan exit
BE56-	A9 10	LDA	#\$10	Was de diskette Write Protected?!
BE58-	B0 EE	BCS	\$BE48	Ja.

BE5A-	48	PHA		MYSEEK subroutine
BE5B-	A0 01	LDY	##01	Nodig voor SEEKABS
BE5D-	B1 3C	LDA	(\$3C),Y	Bepaalt het aantal fasen per track!
BE5F-	6A	ROR		Het is mogelijk met verschillende
BE60-	68	PLA		fasen te werken. (Beveiliging)
BE61-	90 08	BCC	\$BE6B	Knoei hier liever niet aan!
BE63-	0A	ASL		
BE64-	20 6B BE	JSR	\$BE6B	
BE67-	4E 7B 04	LSR	\$047B	
BE6A-	60	RTS		
BE6B-	85 2A	STA	\$2A	
BE6D-	20 8E BE	JSR	\$BE8E	
BE70-	B9 7B 04	LDA	\$047B,Y	
BE73-	24 35	BIT	\$35	
BE75-	30 03	BMI	\$BE7A	
BE77-	B9 FB 04	LDA	\$04FB,Y	
BE7A-	8D 7B 04	STA	\$047B	
BE7D-	A5 2A	LDA	\$2A	
BE7F-	24 35	BIT	\$35	
BE81-	30 05	BMI	\$BE8B	
BE83-	99 FB 04	STA	\$04FB,Y	
BE86-	10 03	BPL	\$BE8B	
BE8B-	99 7B 04	STA	\$047B,Y	
BE8B-	4C A0 B9	JMP	\$B9A0	
BE8E-	8A	TXA		Slot info van X naar Y
BE8F-	4A	LSR		
BE90-	4A	LSR		
BE91-	4A	LSR		
BE92-	4A	LSR		
BE93-	A8	TAY		
BE94-	60	RTS		-----
BE95-	48	PHA		Set track
BE96-	A0 02	LDY	##02	
BE98-	B1 48	LDA	(\$48),Y	
BE9A-	6A	ROR		
BE9B-	66 35	ROR	\$35	
BE9D-	20 8E BE	JSR	\$BE8E	
BEA0-	68	PLA		
BEA1-	0A	ASL		
BEA2-	24 35	BIT	\$35	
BEA4-	30 05	BMI	\$BEAB	
BEA6-	99 FB 04	STA	\$04FB,Y	
BEA9-	10 03	BPL	\$BEAE	
BEAB-	99 7B 04	STA	\$047B,Y	
BEAE-	60	RTS		-----
BEAF-	A0 03	LDY	##03	INIT Afhandelings Routine
BEB1-	B1 48	LDA	(\$48),Y	
BEB3-	85 41	STA	\$41	
BEB5-	A9 AA	LDA	##AA	
BEB7-	85 3E	STA	\$3E	
BEB9-	A0 56	LDY	##56	
BEBB-	A9 00	LDA	##00	Haal volumenummer uit IOB
BEBD-	85 44	STA	\$44	en maak de buffers 0
BEBF-	99 FF BB	STA	\$BBFF,Y	
BEC2-	88	DEY		
BEC3-	D0 FA	BNE	\$BEBF	
BEC5-	99 00 BB	STA	\$BB00,Y	
BEC8-	88	DEY		
BEC9-	D0 FA	BNE	\$BEC5	
BECB-	A9 50	LDA	##50	
BECD-	20 95 BE	JSR	\$BE95	
BED0-	A9 28	LDA	##28	Aantal synobytes tussen sect.

BED2-	85 45	STA	\$45
BED4-	A5 44	LDA	\$44
BED6-	20 5A BE	JSR	\$BE5A
BED9-	20 0D BF	JSR	\$BF0D
BEDC-	A9 08	LDA	#\$08
BEDE-	B0 24	BCS	\$BF04
BEE0-	A9 30	LDA	#\$30
BEE2-	8D 78 05	STA	\$0578
BEE5-	38	SEC	
BEE6-	CE 78 05	DEC	\$0578
BEE9-	F0 19	BEQ	\$BF04
BEEB-	20 44 B9	JSR	\$B944
BEEE-	B0 F5	BCS	\$BEE5
BEF0-	A5 2D	LDA	\$2D
BEF2-	D0 F1	BNE	\$BEE5
BEF4-	20 DC B8	JSR	\$B8DC
BEF7-	B0 EC	BCS	\$BEE5
BEF9-	E6 44	INC	\$44
BEFB-	A5 44	LDA	\$44
BEFD- ✓	C9 23	CMP	#\$23
BEFF-	90 D3	BCC	\$BED4
BF01-	18	CLC	
BF02-	90 05	BCC	\$BF09
BF04-	A0 0D	LDY	#\$0D
BF06-	91 48	STA	(\$48),Y
BF08-	38	SEC	
BF09-	BD 88 C0	LDA	\$C088,X
BF0C-	60	RTS	
BF0D-	A9 00	LDA	#\$00
BF0F-	85 3F	STA	\$3F
BF11-	A0 80	LDY	#\$80
BF13-	D0 02	BNE	\$BF17
BF15-	A4 45	LDY	\$45
BF17-	20 56 BC	JSR	\$BC56
BF1A-	B0 6B	BCS	\$BF87
BF1C-	20 2A B8	JSR	\$B82A
BF1F-	B0 66	BCS	\$BF87
BF21-	E6 3F	INC	\$3F
BF23-	A5 3F	LDA	\$3F
BF25-	C9 10	CMP	#\$10
BF27-	90 EC	BCC	\$BF15
BF29-	A0 0F	LDY	#\$0F
BF2B-	84 3F	STY	\$3F
BF2D-	A9 30	LDA	#\$30
BF2F-	8D 78 05	STA	\$0578
BF32-	99 A8 BF	STA	\$BFA8,Y
BF35-	88	DEY	
BF36-	10 FA	BPL	\$BF32
BF38-	A4 45	LDY	\$45
BF3A-	20 87 BF	JSR	\$BF87
BF3D-	20 87 BF	JSR	\$BF87
BF40-	20 87 BF	JSR	\$BF87
BF43-	48	PHA	
BF44-	68	PLA	
BF45-	EA	NOP	
BF46-	88	DEY	
BF47-	D0 F1	BNE	\$BF3A
BF49-	20 44 B9	JSR	\$B944
BF4C-	B0 23	BCS	\$BF71
BF4E-	A5 2D	LDA	\$2D
BF50-	F0 15	BEQ	\$BF67
BF52-	A9 10	LDA	#\$10

Geef 48 retries na INIT

Controleer track 0

✓ Doe 35 tracks. Kan aangepast worden tot 36, soms meer! Zie ook elders.

Formateer de track

BF54-	C5 45	CMP	\$45
BF56-	A5 45	LDA	\$45
BF58-	E9 01	SBC	#\$01
BF5A-	B5 45	STA	\$45
BF5C-	C9 05	CMP	#\$05
BF5E-	B0 11	BCS	\$BF71
BF60-	38	SEC	
BF61-	60	RTS	
BF62-	20 44 B9	JSR	\$B944
BF65-	B0 05	BCS	\$BF6C
BF67-	20 DC B8	JSR	\$B8DC
BF6A-	90 1C	BCC	\$BF88
BF6C-	CE 78 05	DEC	\$0578
BF6F-	D0 F1	BNE	\$BF62
BF71-	20 44 B9	JSR	\$B944
BF74-	B0 0B	BCS	\$BF81
BF76-	A5 2D	LDA	\$2D
BF78-	C9 0F	CMP	#\$0F
BF7A-	D0 05	BNE	\$BF81
BF7C-	20 DC B8	JSR	\$B8DC
BF7F-	90 8C	BCC	\$BF0D
BF81-	CE 78 05	DEC	\$0578
BF84-	D0 EB	BNE	\$BF71
BF86-	38	SEC	
BF87-	60	RTS	
BF88-	A4 2D	LDY	\$2D
BF8A-	B9 AB BF	LDA	\$BFAB,Y
BF8D-	30 DD	BMI	\$BF6C
BF8F-	A9 FF	LDA	#\$FF
BF91-	99 AB BF	STA	\$BFAB,Y
BF94-	C6 3F	DEC	\$3F
BF96-	10 CA	BPL	\$BF62
BF98-	A5 44	LDA	\$44
BF9A-	D0 0A	BNE	\$BFA6
BF9C-	A5 45	LDA	\$45
BF9E-	C9 10	CMP	#\$10
BFA0-	90 E5	BCC	\$BF87
BFA2-	C6 45	DEC	\$45
BFA4-	C6 45	DEC	\$45
BFA6-	18	CLC	
BFA7-	60	RTS	
BFA8-	FF	???	
BFA9-	FF	???	
BFAA-	FF	???	
BFAB-	FF	???	
BFAC-	FF	???	
BFAD-	FF	???	
BFAE-	FF	???	
BFAF-	FF	???	
BFB0-	FF	???	
BFB1-	FF	???	
BFB2-	FF	???	
BFB3-	FF	???	
BFB4-	FF	???	
BFB5-	FF	???	
BFB6-	FF	???	
BFB7-	FF	???	
BFB8-	00	BRK	
BFB9-	0D 0B 09	ORA	\$090B
BFBC-	07	???	
BFBD-	05 03	ORA	\$03
BFBF-	01 0E	ORA	(\$0E,X)

Verifieer de track

Sector maproutine.
Lees 16 sectoren van juist
geformateerde track.

Sector Map Routine
Markeert de sector Init Map
na verificering van een sector.

Sector Init Map
Eerst \$30, na formatteren \$FF

Sector Interleaving Tabel
Verweeft de sectoren opdat snelle
data transfer mogelijk is.

BFC1-	0C	???	
BFC2-	0A	ASL	
BFC3-	08	PHP	
BFC4-	06 04	ASL	\$04
BFC6-	02	???	
BFC7-	0F	???	
BFC8-	20 93 FE	JSR	\$FE93
BFCB-	AD 81 C0	LDA	\$C081
BFCE-	AD 81 C0	LDA	\$C081
BFD1-	A9 00	LDA	#\$00
BFD3-	8D 00 E0	STA	\$E000
BFD6-	4C 44 B7	JMP	\$B744
BFD9-	00	BRK	
BFDA-	00	BRK	
BFDB-	00	BRK	
BFDC-	8D 63 AA	STA	\$AA63
BFDF-	8D 70 AA	STA	\$AA70
BFE2-	8D 71 AA	STA	\$AA71
BFE5-	60	RTS	
BFE6-	20 5B A7	JSR	\$A75B
BFE9-	8C B7 AA	STY	\$AAB7
BFEC-	60	RTS	
BFED-	20 7E AE	JSR	\$AE7E
BFF0-	AE 9B B3	LDX	\$B39B
BFF3-	9A	TXS	
BFF4-	20 16 A3	JSR	\$A316
BFF7-	BA	TSX	
BFF8-	8E 9B B3	STX	\$B39B
BFFB-	A9 09	LDA	#\$09
BFFD-	4C 85 B3	JMP	\$B385
C000-	13	???	
C001-	13	???	
C002-	D3	???	
C003-	D3	???	
C004-	93	???	
C005-	53	???	

Patches***

Set video

Zeer ongelukkige patch, die telkens opnieuw de Language Card laat laden. BFD3: EA EA EA vrij

Zie A0E2.
Drie vervalwaarden.

Van A6D5.

Van B377
Save f.m. werkruimte.
Restore stack
Close files

Save stack
Run niet onderbroken.
DISK FULL ERROR

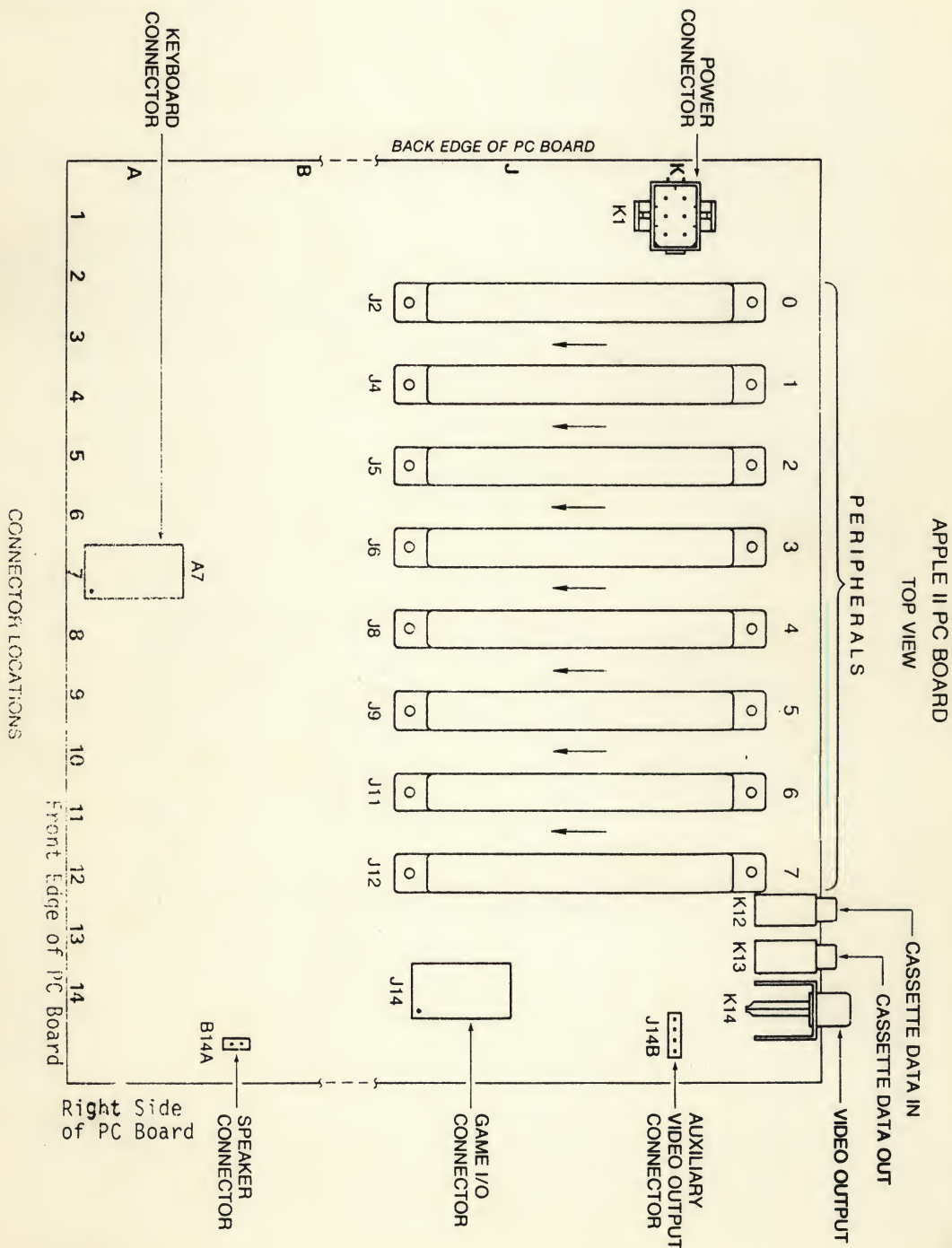
-einde DISASM/82/vK-

A\$9D00/84
DOS A\$ 9600 L\$2A00 V3

HOOFDSTUK .. INTERFACING EN I/O

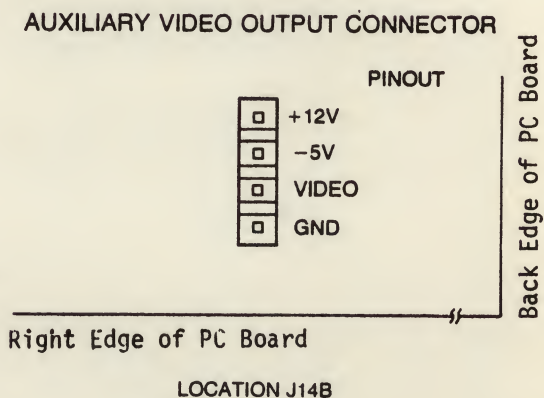
1. Overzicht

Het moederbord van de APPLE II is voorzien van een aantal connectors, die verbinding met andere hardware mogelijk maakt. In figuur A is de plaats van deze connectors aangegeven.



2. Video

Een van de meest prominente hardware-onderdelen bij de APPLE is de videomonitor. De video output K14, een standaard RCA phone jack, levert een signaal, dat geschikt is voor elke "close circuit" of "studio" televisie-monitor. Een monitor is een televisie toestel zonder ontvangstinrichting. Voor aansluiting van de APPLE II op een gewoon TV-ontvangsttoestel is het noodzakelijk, dat een RF-modulator wordt tussengeschakeld. Daarmee kan het signaal direkt worden toegevoerd aan de antenne-ingang. De modulator wordt aangesloten op de speciale video-pin, die zich vanaf het keyboard gezien, juist rechts boven de game-I/O bevindt. Voor toepassing van een kleurentelevisie zijn verdergaande hardware applicaties gewenst. Het door de APPLE II aangeboden signaal is NTSC compatibel positief composite video. Zwartniveau .75 Volt, witniveau 2.0 Volt, sync tip 0 Volt. Niet beschermd tegen kortsluiting. Het is daarom nodig een NTSC monitor te gebruiken, waarbij het -afhankelijk van de versie van het moederbord- bovendien voor kan komen, dat het oscillatorkristal van de computer moet worden aangepast. Een acceptabel alternatief is een z.g. PAL-kleurkaart, die in feite een kleur RF-modulator is. Men moet evenwel bedenken, dat de contourscherpte en kleurzuiverheid van deze applicatie te wensen overlaat. Beter is een z.g. RGB-kaart, die uitsluitend toegepast kan worden met PAL/SECAM kleurmonitoren, of een moderne kleurentelevisie met video ingang. Je verliest met deze kaarten en de speciale modulator een Slot.



Er is op het moederbord nog een derde video-aansluiting te vinden. Deze bestaat uit een Molex KK100 aansluiting (J14b).

extra video output connector met vier square-pin
connectors J14b

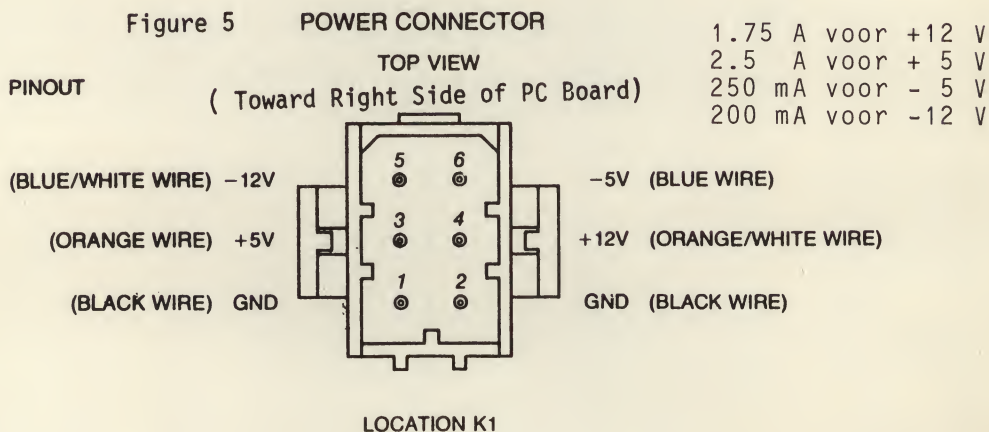
pin	naam	omschrijving
1	massa	0 Volt syteem massaleiding
2	video	NTSC compatibel, positief composite video. DC gekoppelde emittervolger-uitgang; niet kortsluitvast. Sync tip 0 Volt zwartniv. 0.75 V. en witniv. 2.0 V. over 470 ohm. Niveau niet instelbaar.
3	power+	+12 volt uit voeding
4	power-	-5 volt uit voeding

Op het moederbord bevinden zich nabij de video-pin 2
potmeters, waarmee kleur- en videosignaal niveau's kunnen
worden ingesteld binnen een range van 0 tot 1 Volt.

3. Voeding

De APPLE II beschikt over een z.g. switching power supply. Dit type voeding heeft het voordeel niet te zijn uitgerust met veel warmte dissiperende transformatoren. De voeding is voorzien van een netfilter. Storingen uit het lichtnet dringen niet door in de computer. Om dit goed te laten functioneren, alsmede voor je eigen veiligheid, MOET je een stekkervoorziening met randaarde gebruiken. De voedingseenheid mag uitsluitend door deskundige technici worden geopend. In de voeding komen hoge spanningen voor. Er is een handige voorziening aangebracht, om kortsluitingen te onderkennen en de voeding te beschermen.

In verband met de grote hoeveelheid peripherals voor de APPLE Interface Slots is het raadzaam de maximum-belasting in het oog te houden. Deze mag voor alle toegevoegde interfacekaarten niet meer bedragen dan:



De werkt temperatuur voor de voeding is ca. 60 graden Celcius.

Wanneer de APPLE door veel interfacekaarten wordt aangesproken, is het zeer aan te bevelen een speciale ventilator in te bouwen. Deze ventilator dient bij voorkeur op 220 V te zijn aangesloten.

Wordt de binnentemperatuur te hoog, dan kan de bedrijfszekerheid van het systeem in het gedrang komen. Ook bij een buitenlucht temperatuur van meer dan 25 graden Celcius kunnen zich problemen voordoen, wanneer gewerkt wordt met een aanzienlijke interface- belasting in een slecht circulerende omgeving.

De verbinding tussen voeding en moederbord is een blokconnector op lokatie K1.

4. Cassetterecorder

Je kunt een eenvoudige cassetterecorder met (automatische) opname regeling gebruiken als (extra) opslagmedium voor programma's en gegevens. Er mag geen gebruik worden gemaakt van de z.g. lijn- ingang van de cassetterecorder, maar van de microfooningang en de koptelefoonuitgang.

Achter de computer bevindt zich naast de video aansluiting een tweetal cassette data pluggen. "Cassette Data In" (K12) moet worden aangesloten op de koptelefoon of monitoruitgang van de recorder.

V in= 1 V pp nominaal Z in= 12 KOhm

"Cassette Data Out" (K13) aansluiten op de microfooningang.

V in= 25mV in 100 Ohm Z out= 100 Ohm

Het gebruik van de cassetterecorder vereist enige handigheid. Als de opname te zwak is, zal de signaal-ruis verhouding bij de weer- gave te laag zijn. Is de opname te sterk, dan volgt daaruit enige vervorming van het signaal. Luister daarom eens naar het geluid van je APPLE. Probeer uit te vinden, wat de ideale opname en weergave stand is van de recorder. Zet om te beginnen het opnameniveau op 60% met zo mogelijk 80% hoog bij. Plaats een C60 cassettebandje in de recorder. Afhankelijk van de opname moet je LOAD, RECALL, of *R tikken. Zie de bespreking van de beschikbare softwaretechnieken. Start de recorder en druk Return. Het volgende zal dan gebeuren:

- a) tweemaal een "biep" toontje en de cursor keert terug.
- b) foutmelding ?SYNTAX ERROR volgt
- c) er gebeurt niets - cursor blijft weg
- d) een "biep" en daarna niets meer
- e) er verschijnt ERR, soms gevolgd door een "biep"

De onder a) genoemde situatie is normaal. Spoel het bandje na het gebruik weer terug.

De onder b) genoemde situatie duidt erop, dat voor het begin van de inleidende fluittoon werd ingeschakeld.

Begin opnieuw.

De onder c) en d) vermelde toestand duidt op een te laag niveau. Los de "hang"-toestand op door Reset.

De onder e) genoemde situatie ontstaat door een te hoog niveau. Zet de volumeknop iets minder hoog en begin opnieuw.

Als het laden na herhaalde pogingen, zonodig met een kort Applesoft programmaatje, maar niet wil lukken, zijn daarvoor verschillende oorzaken aan te wijzen. Soms is het programma te groot (voor b.v. 16K computers). Mogelijk is LOMEM: of HIMEM: zodanig ingesteld, dat het programma niet ingeladen kan worden. Tenslotte kunnen verkeerd afgestelde, of versleten en vervuilde koppen van de recorder oorzaak zijn van veel moeilijkheden. Ook kan de gebruikte tape minder geschikt zijn (C90 type).

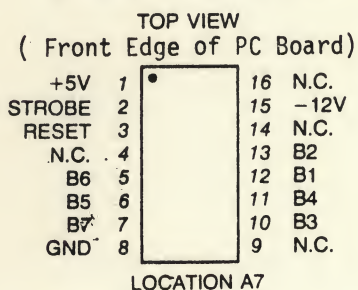
Het meest voorkomend probleem is de onbewuste verwisseling van de opname en weergave plugjes. Lukt het desondanks niet, neem dan gerust contact op met uw computerleverancier. Zelden ligt het probleem aan uw APPLE!

5. Keyboard

Het APPLE toetsenbord (keyboard) is een QWERTY configuratie, gegroepeerd rond een MM5740 monolitische keyboard decoder. De inputlijnen van deze ROM zijn verbonden met de keyboard schakelmatrix. Via een keyboard connector is het toetsenbord met het moederbord verbonden. De keyboard decoder bekijkt met zeer hoge snelheid de keyboardmatrix, waarop de toetsen zijn gemonteerd. Wordt een toets ingedrukt, dan ontstaat een gecodeerd signaal. De REPT toets is verbonden met een 555 timerchip op het keyboard. Door de weerstand van 220 KOhm (R3) te verlagen, ontstaat een hogere REPT snelheid. De lijnen 5-7, 10-13 van de keyboard connector vormen de ASCII keyboard data input in 7 bits formaat.

Naast de keyboard connector op het moederbord bevindt zich de z.g. karakter generator. Dit is een chip van het type 2513, die in moederbord versie 7 ook met ROM SPCL wordt aangeduid. Deze ROM is er de oorzaak van, dat geen kleine letters door de APPLE II worden afgebeeld. Het behoeft geen betoog, dat m.n. de ROM SPCL (pin compatibel met een EPROM 2716), door velen is verbeterd! De auteurs van dit boek kunnen belangstellenden hiermee eventueel helpen.

KEYBOARD CONNECTOR

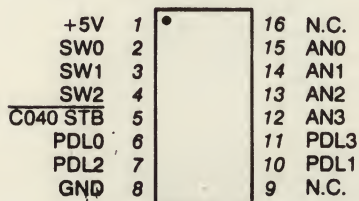


6. Game I/O

Op positie J14 bevindt zich de z.g. Spelletjes- of Game I/O connector. Deze kan worden gebruikt als aansluiting voor kleine lampjes, LED's, lichtpen, game-paddles etc. De connector omvat een 16 pins IC socket, waarvan de pennen de volgende werking hebben:

GAME I/O CONNECTOR

(Front ^{TOP VIEW} Edge of PC Board



LOCATION J14

AN 0-AN 3 4 outputlijnen. Elke lijn is een 74LS TTL output.

Er zijn 8 adressen voor besturing: C058/C05F

C040 STB Strobe output. Eeveneens een 74LS TTL output
De lijn gaat laag gedurende ph.2 van
READ/WRITE naar de adressen C040/C04F

GND Massa. 0 Volt aan voeding

NC niet aangesloten (not connected)

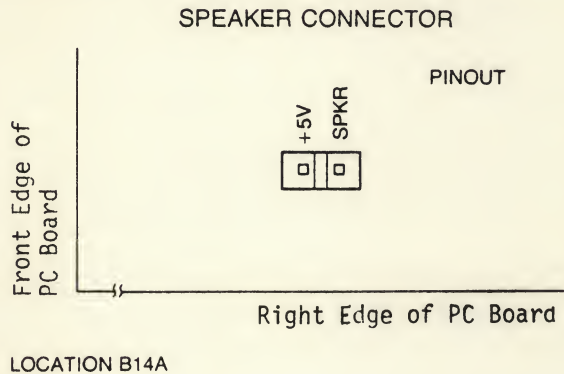
PDL 0-PDL 3 Paddle control lijnen. Hierop kan een
variabele
weerstand van 150K0hm worden aangesloten. De
andere zijde van de weerstand ligt aan +5
Volt. Een ingebouwde weerstand van 100 Ohm
begrenst de stroom. Aflezing door PDL
commando.

SW 0-SW 2 Switch inputs. Elke lijn is een 74LS TTL
ingang.
De lijnen zijn te testen via adressen C061/63
of C069/6B. Pushbutton aansluiting.

+5 V Positieve spanning van + 5 Volt; max. 100 mA.

7. Geluid

Op het moederbord is een connector aanwezig voor aansluiting van een luidspreker. In de APPLE II is een kleine LS aangebracht, die aangesproken kan worden via adres -16336/C030. De aandrijving geschiedt door een Darlington circuit. De lijn levert 0.5 Watt aan 8 Ohm. De speakerconnector bevat 2 aansluitingen:



8. De Periferal Connectors (of Slots)

Op het APPLE PC-bord zijn 8 periferal connectors opgenomen, die men wel aanduidt met het woord "Slot". Ze zijn bedoeld om op eenvoudige wijze accessoires met de APPLE computer te verbinden. Daartoe worden gerekend printers, diskdrives, diverse interface besturingskaarten. De Slots zijn 50 pins "card edge" connectors van het type Winchester 2HW25C0-111. De betekenis van de pennen is als volgt:

- Pin 1 Select I/O Voor elk slot zijn 16 adressen gereserveerd.
Wordt daarop gelezen of geschreven, dan gaat
- pin 41 Gedurende ph 2. 500 nS laag. Er kunnen max. 5 standaard TTL load worden gedreven.
- Pin 2/17 A0/A15 16 bits adresbus. De adressen van de 6502 processor zijn 30 nS na ph 2. stabiel. Elke lijn kan 5 LS TTL loads sturen.
- Pin 18 R/WR Gebufferde Read Write lijn van de 6502. Isde lijn hoog, dan is de Read cyclus aan de gang. Max. 16 TTL loads.
- Pin 19 SYNC Uitsluitend op Slot 7 !!! Gekoppeld aan de video-sync timing generator.
- Pin 20 I/O STR In/Out Strobe. Wordt laag gedurende ph 2. bij een Read/Write op een van de adressen C800-CFFF. Max. 4 TTL loads
- Pin 21 RDY Ready lijn naar 6502.
Deze lijn mag alleen van niveau veranderen gedurende ph 1. Gaat de lijn laag, dan stopt de Mpu bij de volgende Read cyclus. Deze lijn heeft een 3K0hm pull up weerstand naar +5 V. en moet gedreven worden met een open collector uitgang.
- Pin 22 DMA Direct Memory Access daisy chain output Max. 4 TTL loads. Lijn wordt hoog gehouden door een 3 K0hm weerstand aan +5 V.
- Pin 23 INT OUT Interrupt daisy chain naar device met lagere prioriteit. Max. 4 TTL loads.
Meestal gecombineerd met Pin 28 INT IN
- Pin 24 DMA OUT DMA daisy chain output naar device met lagere prioriteit. Meestal gecombineerd met
- Pin 22 DMA.
- Pin 25 +5 +5 Volt van de power supply. Max. 500 MA voor alle peripherals.
- Pin 26 GND Massa lijn. 0 Volt aan voeding.
- Pin 27 DMA IN DMA daisy chain input voor device met hogere prioriteit. Combineert met Pin 24.
- Pin 28 INT IN Interrupt daisy chain voor device met hogere prioriteit. Combineert met Pin 23.
- Pin 29 NMI Non Maskable Interrupt lijn naar 6502.
Wordt deze lijn laag, dan ontstaat een afhandelingsroutine met een JMP naar \$3FB. De lijn heeft een 3 K0hm pull up weerstand aan +5 V.

Pin 30 IRQ Interrupt Request lijn naar 6502.
 Wordt de lijn laag, dan begint alleen een
 interrupt afhandeling indien de I-flag niet
 geset is. In \$3FE/\$3FF moet het adres
 aanwezig zijn voor de afhandelingsroutine.

Pin 31 RES Reset lijn vanaf keyboard. Kan 2 MOS
 loads sturen per slot.

Pin 32 INH Inhibit lijn. Trekt een device deze
 lijn laag, dan worden alle ROM's op het moederbord
 uitgeschakeld. Lijn blijft hoog over 3 KOhm
 weerstand aan +5 Volt.

Pin 33 -12 -12 Volt van voeding.

Pin 34 - 5 - 5 Volt van voeding.

Pin 35 COLOR Uitsluitend Slot 7 !!! Gekoppeld aan
 de 3.5 MHz color reference video generator.

Pin 36 7M / MHz klok. Max. 2 LS TTL loads

Pin 37 Q3 2 MHz klok, asymmetrisch.

Pin 38 ph 1. Phase 1 klok.

Pin 39 USER 1 Normaal niet in gebruik. Indien laag,
 dan wordt alle I/O adresdecoding opgehouden.

Pin 40 ph 0. Phase 0 klok.

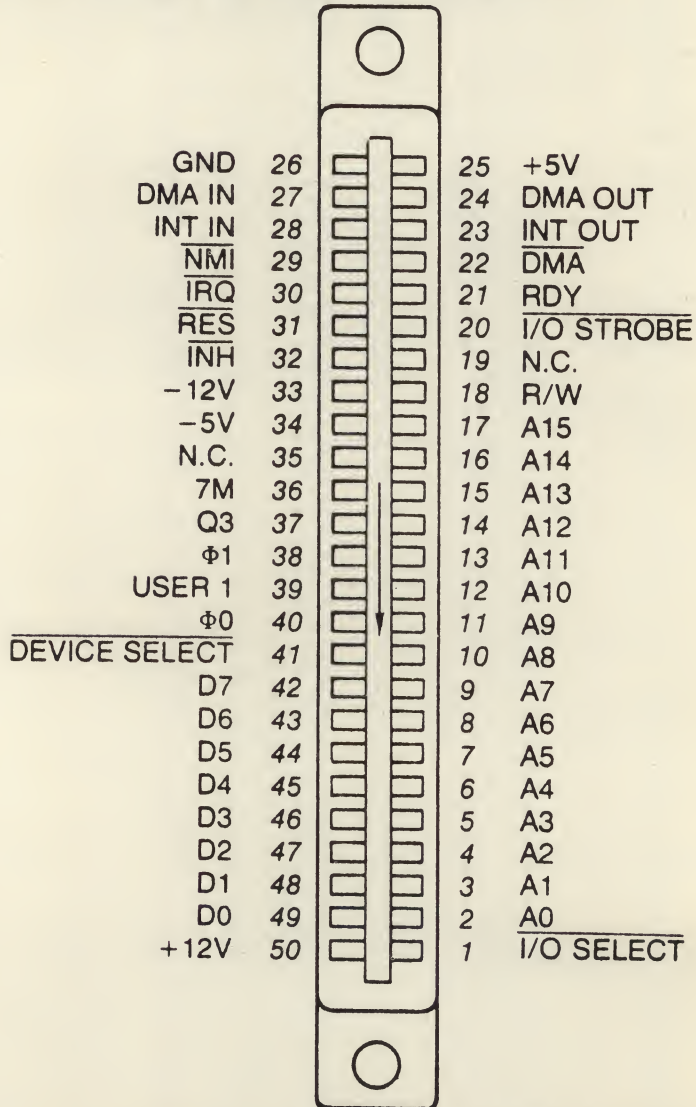
Pin 41 DEV SEL Zie pin 1. Laag wanneer de adresbus
 een adres bevat tussen \$C0s0 en \$C0sF.
 s = Slot + \$8.

Pin 42/49 D0/7 Databus. Dit is een 8 bit systeem.

Pin 50 +12 Volt van voeding.

PERIPHERAL CONNECTORS (EIGHT OF EACH)

TOP VIEW
PINOUT (Back Edge of PC Board)



(Toward Front Edge of PC Board)
LOCATIONS J2 TO J12

9. Nadere beschouwing I/O

Het APPLE II Reference Manual geeft nog uitgebreider weer, welke adressen en toepassingen via de I/O structuur mogelijk zijn. Een veelgehoorde vraag is evenwel: "Wat doe je praktisch gesproken met die uiteenzetting?" Voor de niet hardware georiënteerde gebruiker rijst de vraag, of hij veel mist, door geen kennis te nemen van de I/O mogelijkheden. Het antwoord kan ontkennend en bevestigend luiden. Zeer weinig gebruikers zullen de APPLE toepassen, om eigen hardware ontwikkelingen te testen. Veel gebruikers hebben daarentegen te maken met het gebruik van diskstations, printers en interfacekaarten, zoals een klokkaart of een languagekaart.

Het is onze ervaring, dat veel problemen ontstaan door het overladen van de APPLE-voeding. Door teveel interfaces te gebruiken ontstaan bovendien warmteproblemen, die de bedrijfszekerheid van de computer geweld aandoen.

Het overweldigende potentieel van de APPLE nodigt nu eenmaal uit tot extravagante uitbreidingen. Kunnen we voor de interfacing een praktische standaard vaststellen? We wagen het erop enige tips te geven, die voortkomen uit praktijkervaringen en Amerikaanse gebruiken.

- Slot 0. Integer Basiskaart/Applesoftkaart of Languagekaart.
- Slot 1. Printer interfacekaart.
- Slot 2. Klokkaart (Mountain hardware, of voor wie eens wat anders wil: ComputerWatch)
- Slot 3. 80 koloms kaart, bijv. van Videx, of Omnivision. Helaas veroorzaken deze kaarten nog wel problemen. Ze worden erg heet.
- Slot 4. Z-80 kaart. Een fantastisch compromis.
- Slot 5. Telefoonmodem voor data-networking. Amerikaanse modems werken niet zondermeer feilloos op ons telefoonnet. Let hierop, anders kost het u een hoge rekening.
- Slot 6. Disk II disktestations
- Slot 7. Of: kleurentelevisie PAL/RGB kaart
Of: modulator voor zw/w televisietoestel.

Nuttiger: interface naar 8 inch diskstations (b.v.SVA)

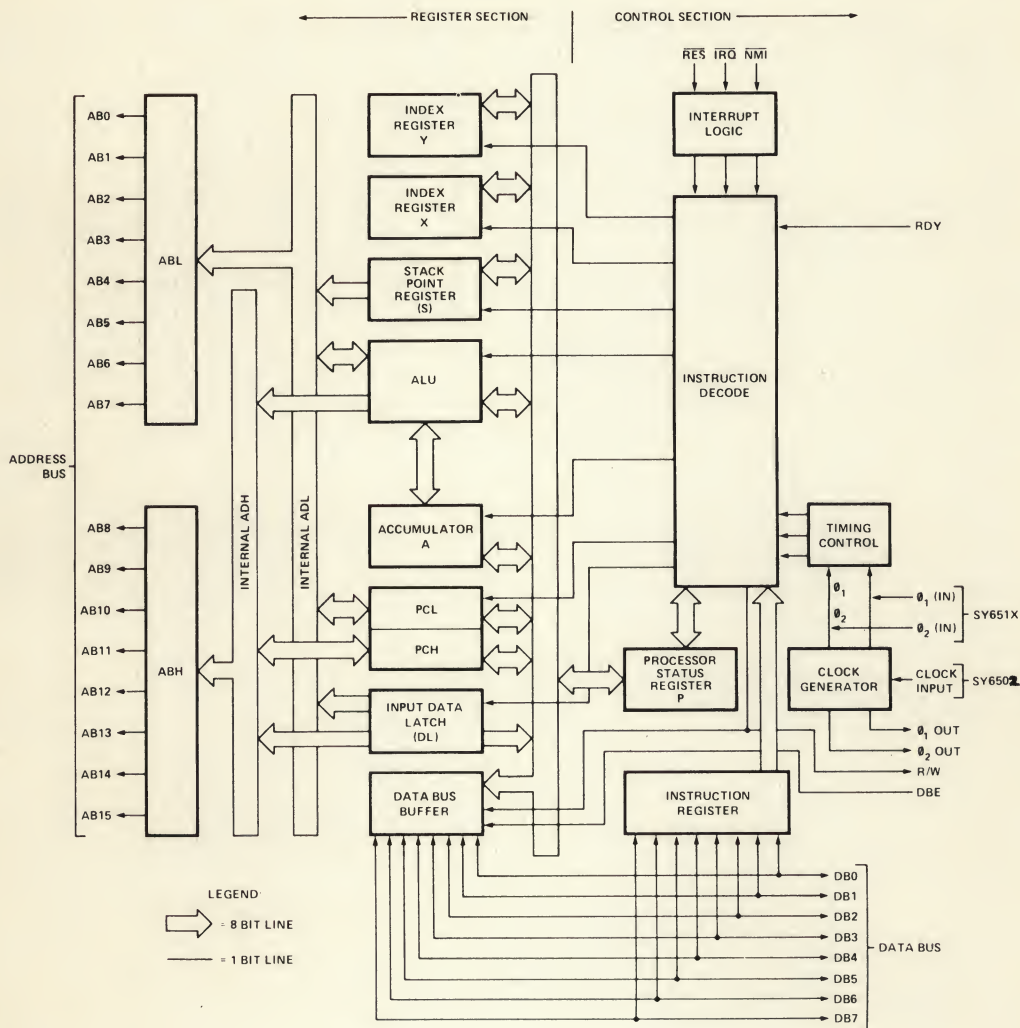
Helaas zijn er relatief veel goedkope namaakkaarten in de handel, die door hobbyisten werden gemaakt. Pas hier voor op! Ze kunnen onderdelen van je computer beschadigen. De meeste software grijpt terug naar deze aansluitingen. Zijn alle slots bezet, dan is extra koeling zondermeer te adviseren. Het verbruik van deze kaarten grenst aan het maximum van de standaard APPLE II voeding. De voeding wordt dan heter, dan gewoonlijk. Veel troubles komen hieruit voort. Koelen is de oplossing. Gebruik een 220 V. koelventilator; via uw leverancier te betrekken.

I/O and ROM Address Detail

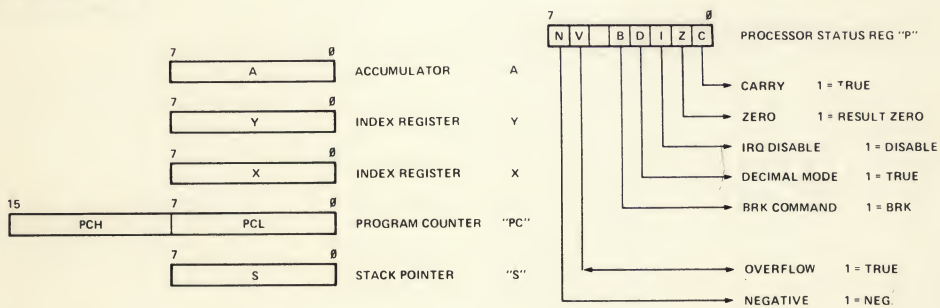
HEX ADDRESS	ASSIGNED FUNCTION	COMMENTS
C00X	Keyboard input.	Keyboard strobe appears in bit 7. ASCII data from keyboard appears in the 7 lower bits.
C01X	Clear keyboard strobe.	
C02X	Toggle cassette output.	
C03X	Toggle speaker output.	
C04X	"C040 STB"	Output strobe to Game I/O connector.
C050	Set graphics mode	
C051	" text "	
C052	Set bottom 4 lines graphics	
C053	" " " " text	
C054	Display primary page	
C055	" secondary page	
C056	Set high res. graphics	
C057	" color "	Annunciator 0 output to Game I/O connector.
C058	Clear "AN0"	
C059	Set "	Annunciator 1 output to Game I/O connector.
C05A	Clear "AN1"	
C05B	Set "	Annunciator 2 output to Game I/O connector.
C05C	Clear "AN2"	
C05D	Set "	Annunciator 3 output to Game I/O connector.
C05E	Clear "AN3"	
C05F	Set "	

HEX ADDRESS	ASSIGNED FUNCTION	COMMENTS
C060/8	Cassette input	State of "Cassette Data In" appears in bit 7.
C061/9	"SW1"	input on State of Switch 1 \wedge Game I/O connector appears in bit 7.
C062/A	"SW2"	State of Switch 2 input on Game I/O connector appears in bit 7.
C063/B	"SW3"	State of Switch 3 input on Game I/O connector appears in bit 7.
C064/C	Paddle 0 timer output	State of timer output for Paddle 0 appears in bit 7.
C065/D	" 1 "	State of timer output for Paddle 1 appears in bit 7.
C066/E	" 2 "	State of timer output for Paddle 2 appears in bit 7.
C067/F	" 3 "	State of timer output for Paddle 3 appears in bit 7.
C07X	"PDL STB"	Triggers paddle timers during ϕ_2 .
C08X	<u>DEVICE SELECT</u> 0	Pin 41 on the selected Peripheral Connector goes low during ϕ_2 .
C09X	" 1	
C0AX	" 2	
C0BX	" 3	
C0CX	" 4	
C0DX	" 5	
C0EX	" 6	
C0FX	" 7	
C10X	" 8	
C11X	" 9	
C12X	" A	Expansion connectors.

HEX ADDRESS	ASSIGNED FUNCTION		COMMENTS
C13X	<u>DEVICE SELECT</u>	B	"
C14X	"	C	"
C15X	"	D	"
C16X	"	E	"
C17X	"	F	"
C1XX	<u>I/O SELECT</u>	1	Pin 1 on the selected Peripheral Connector goes low during ϕ_2 . NOTES: 1. Peripheral Connector 0 does not get this signal. 2. <u>I/O SELECT</u> 1 uses the same addresses as <u>DEVICE SELECT</u> 8-F.
C2XX	"	2	
C3XX	"	3	
C4XX	"	4	
C5XX	"	5	
C6XX	"	6	
C7XX	"	7	
C8XX	"	8, <u>I/O STROBE</u>	Expansion connectors.
C9XX	"	9, "	
CAXX	"	A, "	
CBXX	"	B, "	
CCXX	"	C, "	
CDXX	"	D, "	
CEXX	"	E, "	
CFXX	"	F, "	
D000-D7FF	ROM socket D0		Spare.
D800-DFFF	"	D8	Spare.
E000-E7FF	"	E0	BASIC.
E800-EFFF	"	E8	BASIC.
F000-F7FF	"	F0	1K of BASIC, 1K of utility.
F800-FFFF	"	F8	Monitor.



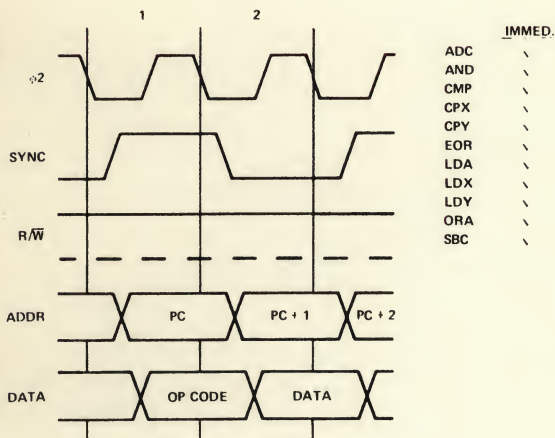
SY6500 Internal Architecture



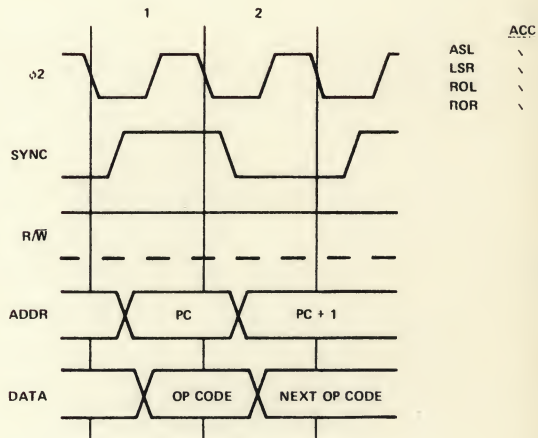
Processor Programming Model

MNE.	IMMED.		ABS		Z-PAGE		ACCUM.		IMPLIED		(IND,X)		(IND,Y)		ZP,X		ABS,X		ABS,Y		REL.		IND.		ZP,Y	
	OP	Fig.	OP	Fig.	OP	Fig.	OP	Fig.	OP	Fig.	OP	Fig.	OP	Fig.	OP	Fig.	OP	Fig.	OP	Fig.	OP	Fig.	OP	Fig.	OP	Fig.
ADC	69	1	6D	6	65	10					61	17	71	18	75	12	7D	6	79	6						
AND	29	1	2D	6	25	10					21	17	31	18	35	12	3D	6	39	6						
ASL			0E	8	06	11	0A	2							16	13	1E	9								
BCC																					90	16				
BCS																					B0	16				
BEQ																					F0	16				
BIT			2C	6	24	10																				
BMI																					30	16				
BNE																					D0	16				
BPL																					10	16				
BRK									00	22																
BVC																					50	16				
BVS																					70	16				
CLO									18	3																
CLD									D8	3																
CLI									58	3																
CLV									B8	3																
CMP	C9	1	CD	6	C5	10					C1	17	D1	18	D5	12	DD	6	D9	6						
CPX	E0	1	EC	6	E4	10																				
CPY	C0	1	CC	6	C4	10																				
DEC			CE	8	C6	11									D6	13	DE	9								
DEX									CA	3																
DEY									88	3																
EOR	49	1	4D	6	45	10					41	17	51	18	55	12	5D	6	59	6						
INC			EE	8	E6	11									F6	13	FE	9								
INX									E8	3																
INY									C8	3																
JMP			4C	15																			6C	14		
JSR			20	20																						
LDA	A9	1	AD	6	A5	10					A1	17	B1	18	B5	12	BD	6	B9	6						
LDX	A2	1	AE	6	A6	10												BE	6					B6	12	
LDY	A0	1	AC	6	A4	10									B4	12	BC	6								
LSR			4E	8	46	11	4A	2							56	13	5E	9								
NOP									EA	3																
ORA	09	1	0D	6	05	10					01	17	11	18	15	12	1D	6	19	6						
PHA									48	4																
PHP									08	4																
PLA									68	5																
PLP									28	5																
ROL			2E	8	26	11	2A	2							36	13	3E	9								
ROR			6E	8	66	11	6A	2							76	13	7E	9								
RTI									40	23																
RTS									60	21																
SBC	E9	1	ED	6	E5	10					E1	17	F1	18	F5	12	FD	6	F9	6						
SEC									38	3																
SED									F8	3																
SEI									78	3																
STA			8D	6	85	10					81	17	91	19	95	12	90	7	99	7						
STX			8E	6	86	10																		96	12	
STY			8C	6	84	10									94	12										
TAX									AA	3																
TAY									A8	3																
TSX									BA	3																
TXA									8A	3																
TXS									9A	3																
TYA									98	3																

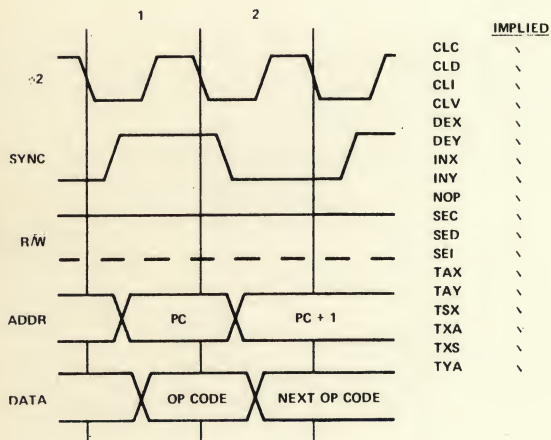
Op Code Reference Chart



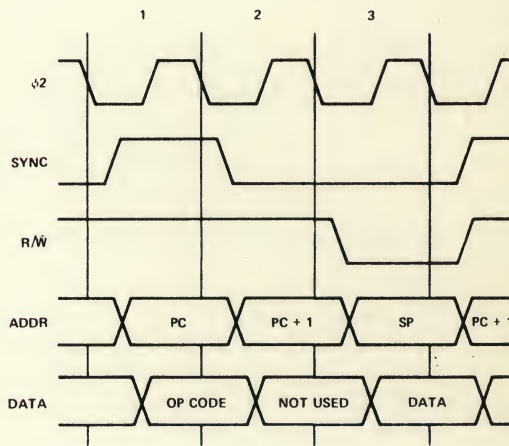
(1) Immediate Addressing



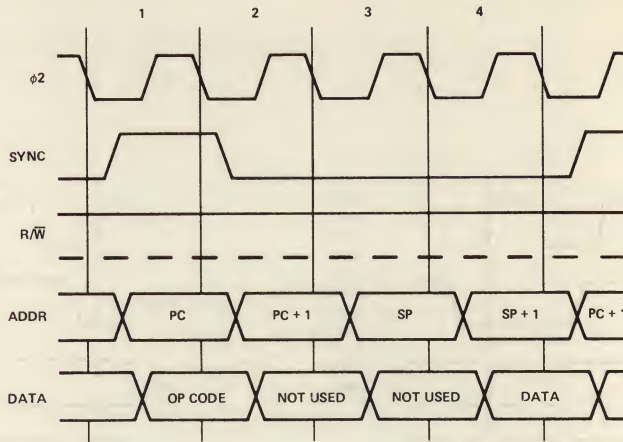
(2) Accumulator



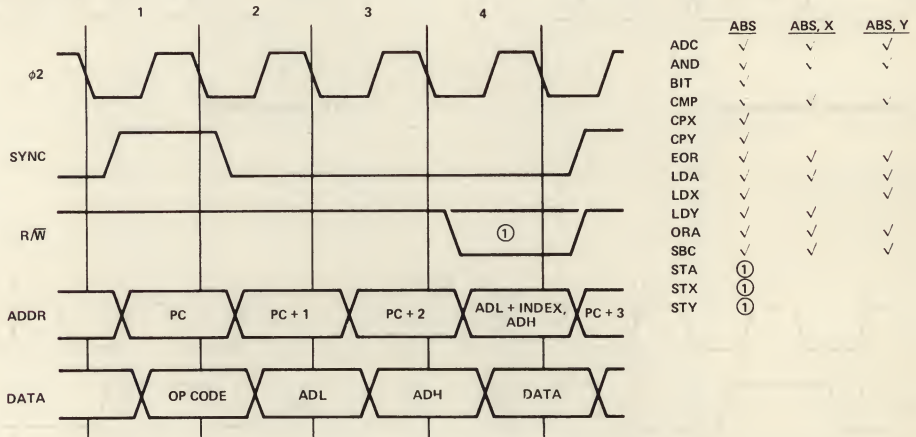
(3) Implied



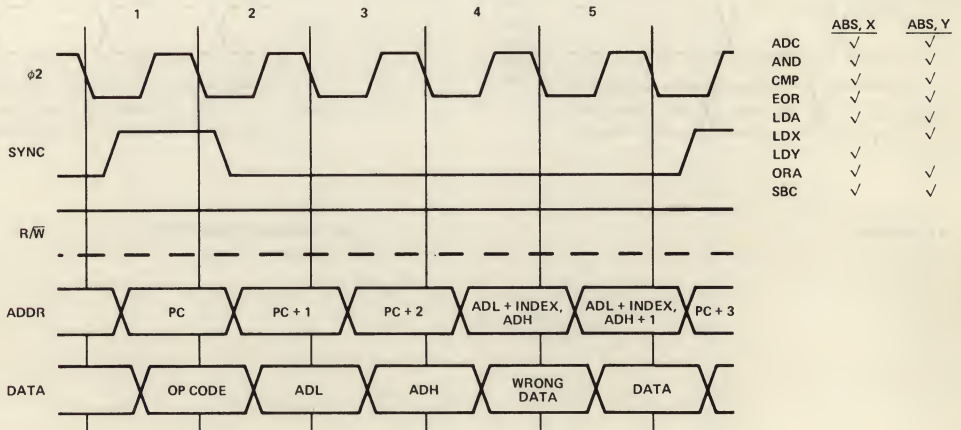
(4) Implied (PHA, PHP)



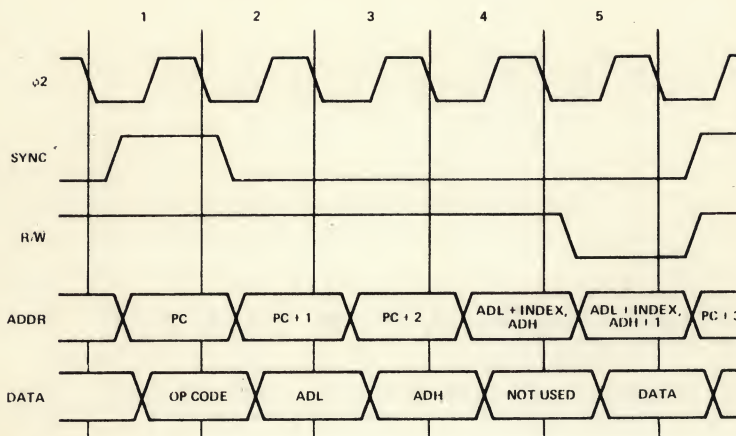
(5) Implied (PLA, PLP)



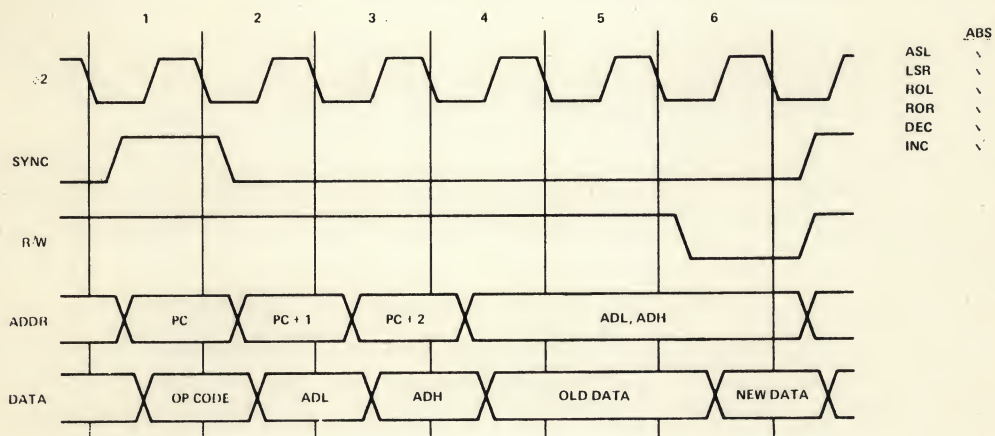
(6a) Absolute and Absolute Indexed (No Page Crossing)



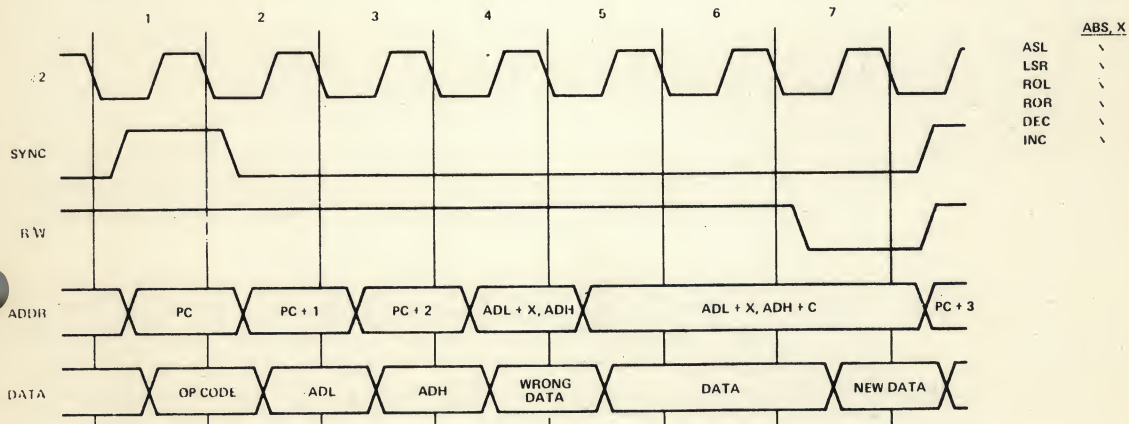
(6b) Absolute Indexed with Page Crossing



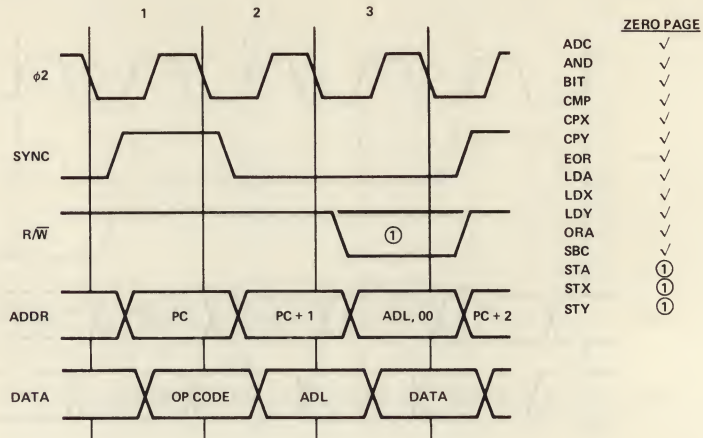
(7) Absolute Indexed with Page Crossing (STA)



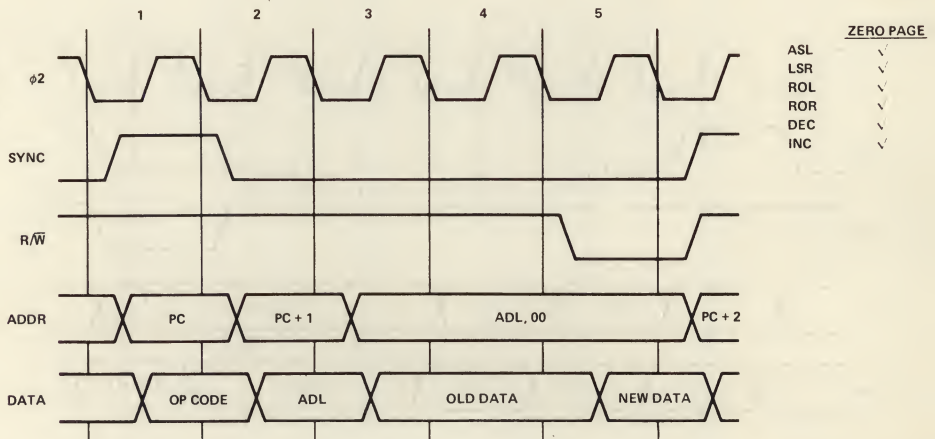
Absolute



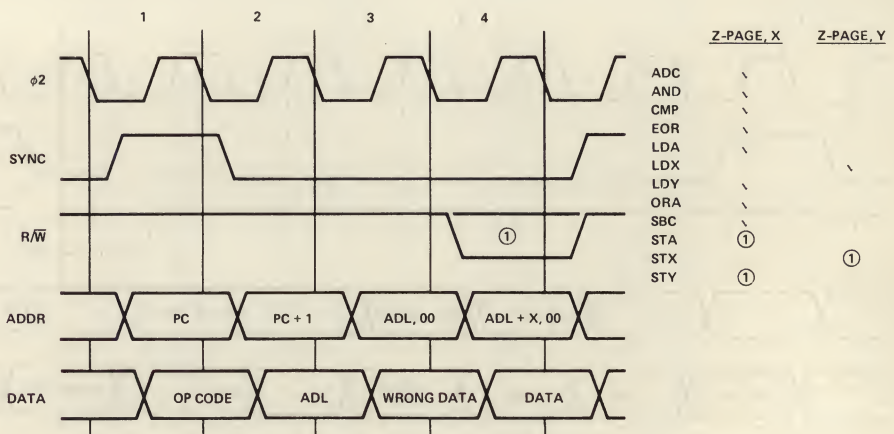
(9) Absolute Indexed



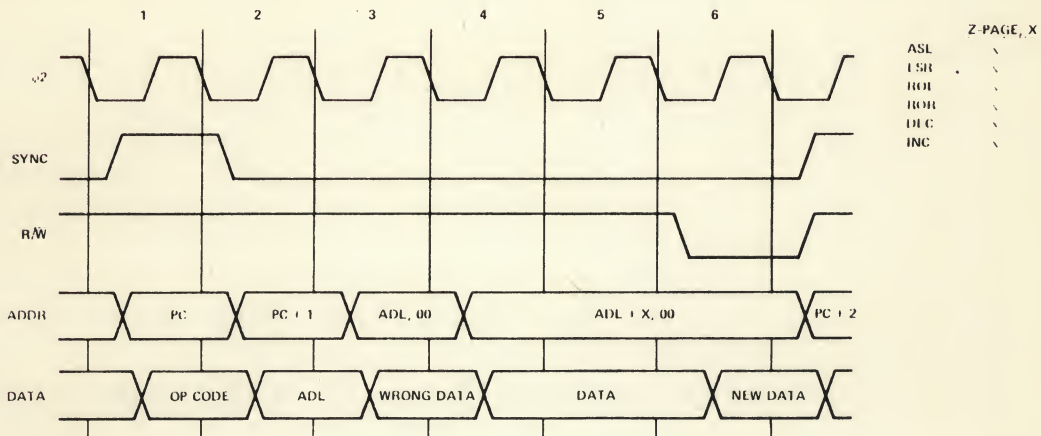
(10) Zero Page



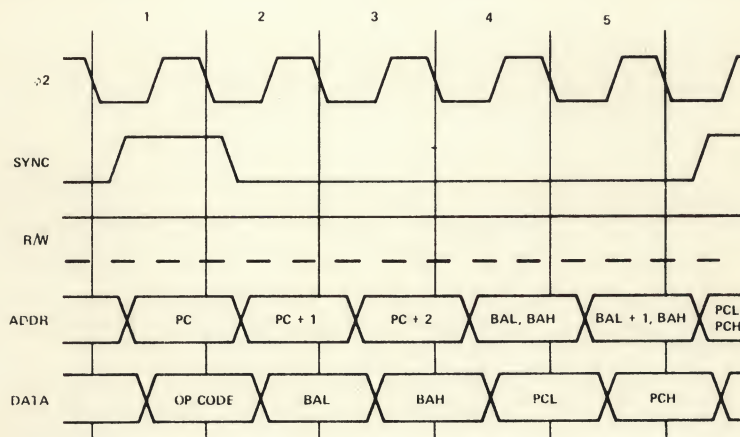
(11) Zero Page



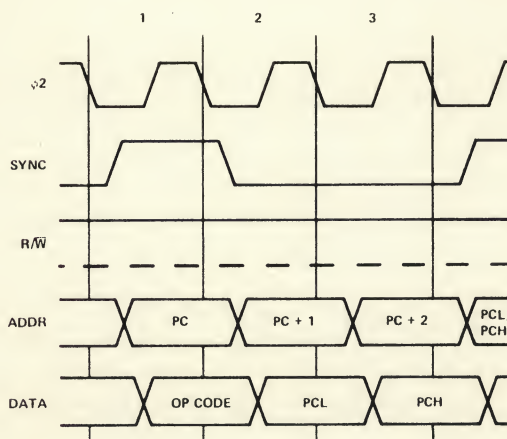
(12) Zero Page Indexed



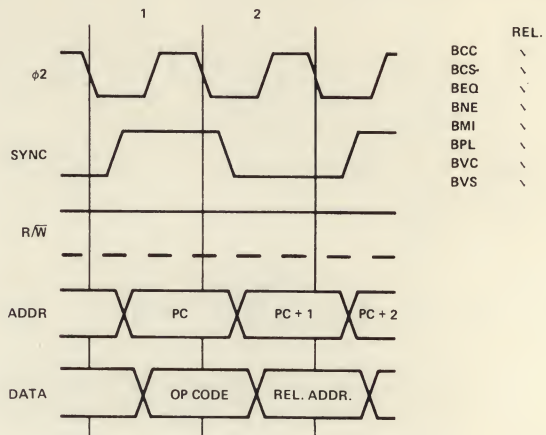
(13) Zero Page Indexed



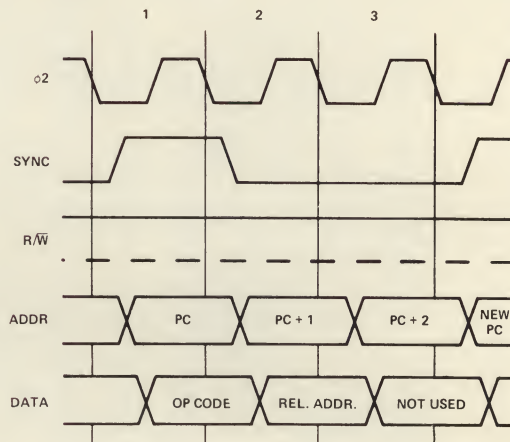
(14) JMP Indirect



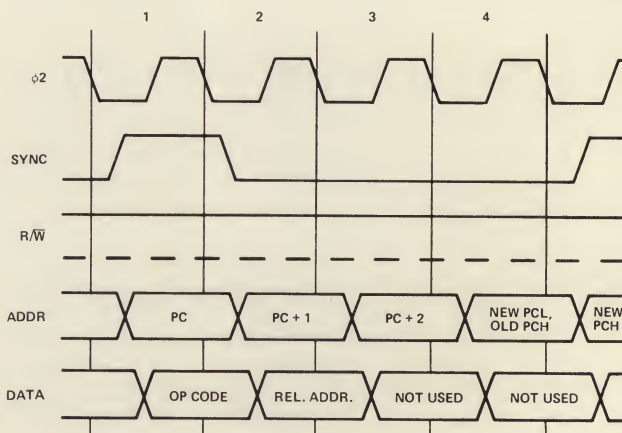
(15) JMP Absolute



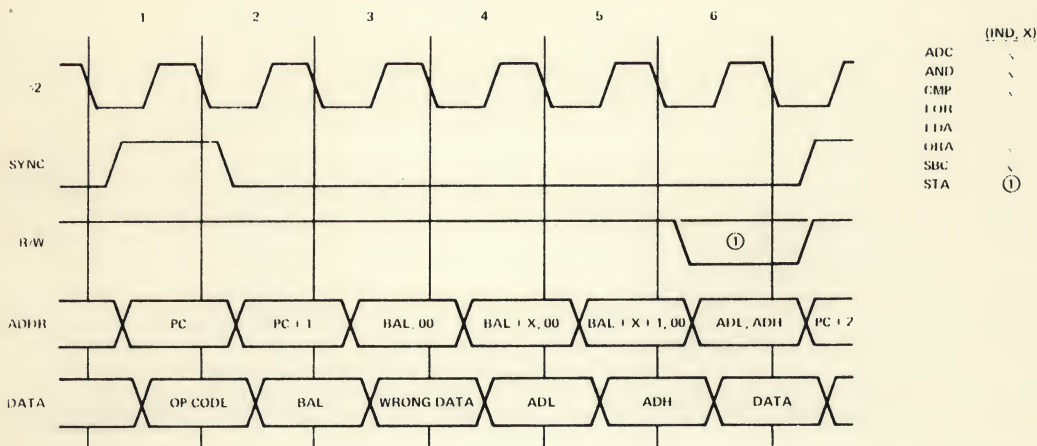
(16a) Relative (Branch Not Taken)



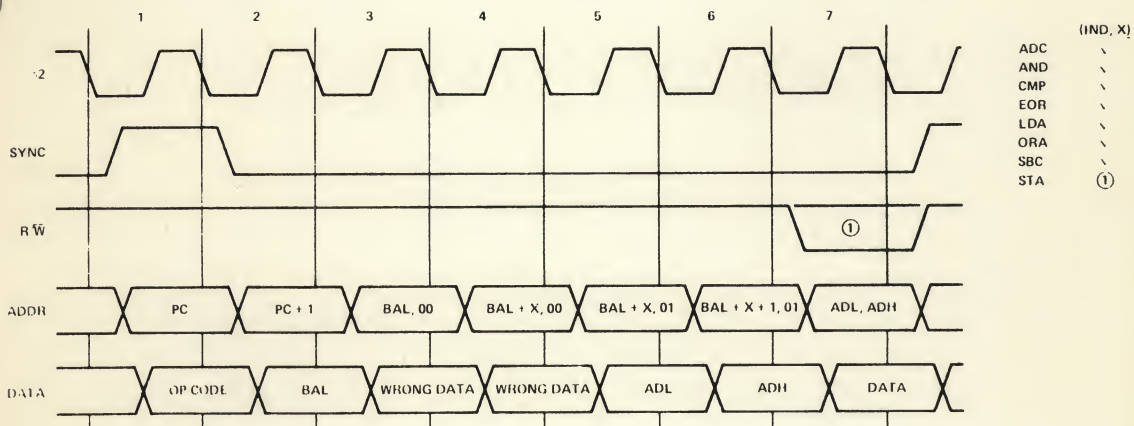
(16b) Relative (No Page Crossing)



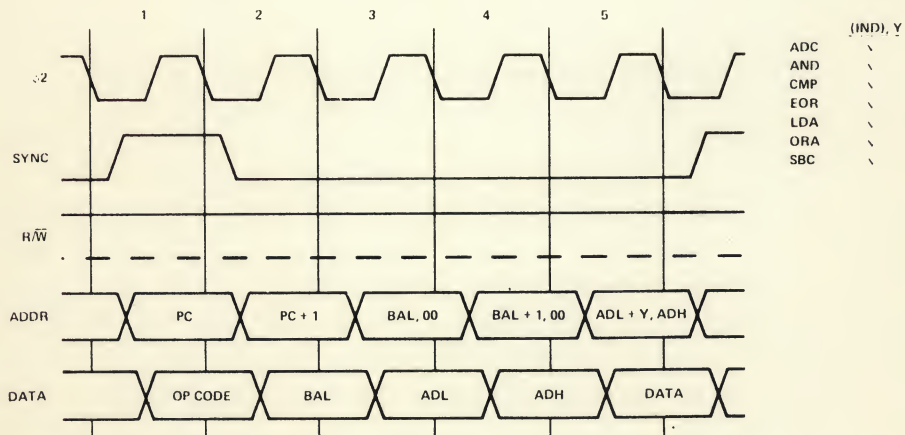
(16c) Relative (With Page Crossing)



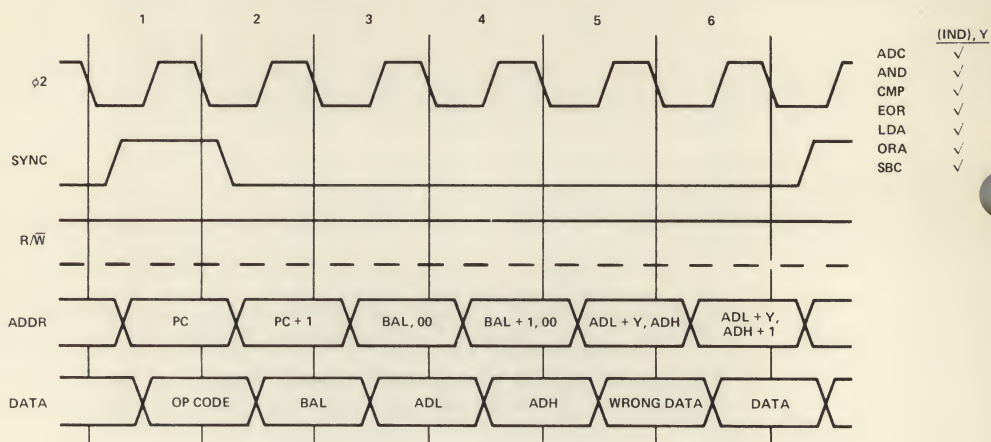
(17a) Indexed Indirect (No Page Crossing)



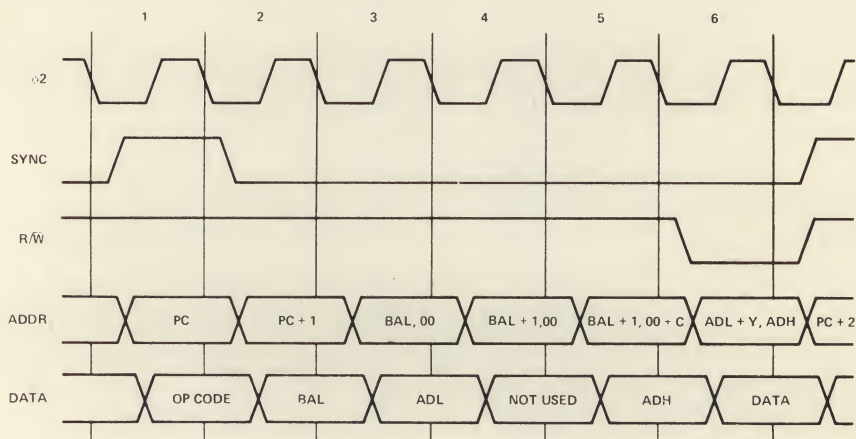
(17b) Indexed Indirect (With Page Crossing)



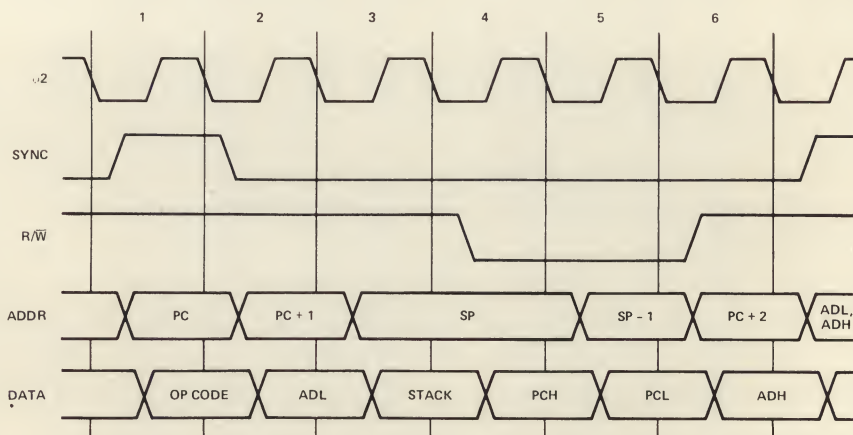
(18a) Indirect Indexed (No Page Crossing)



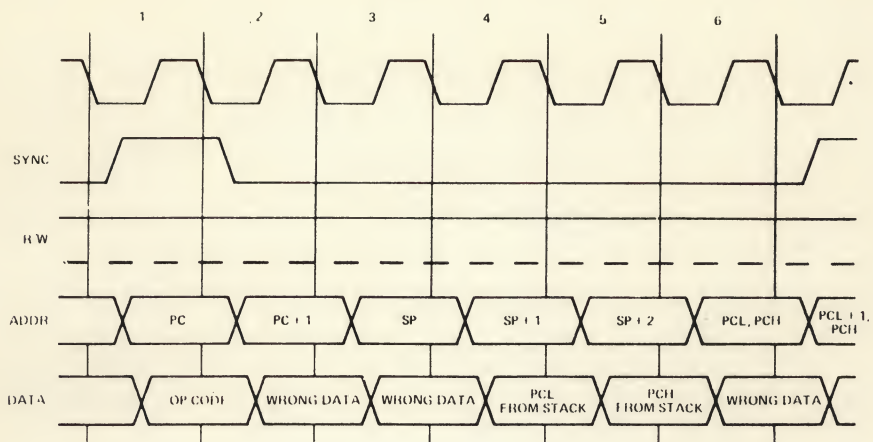
(18b) Indirect Indexed (With Page Crossing)



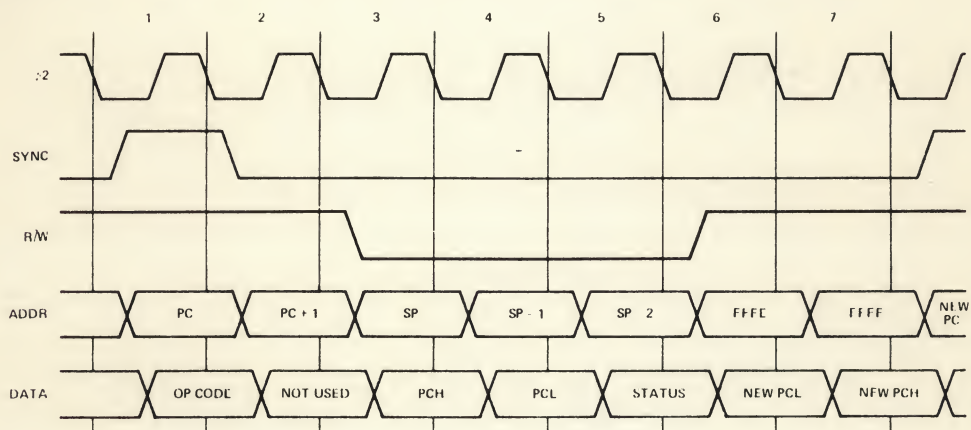
(19) Indirect Indexed (STA)



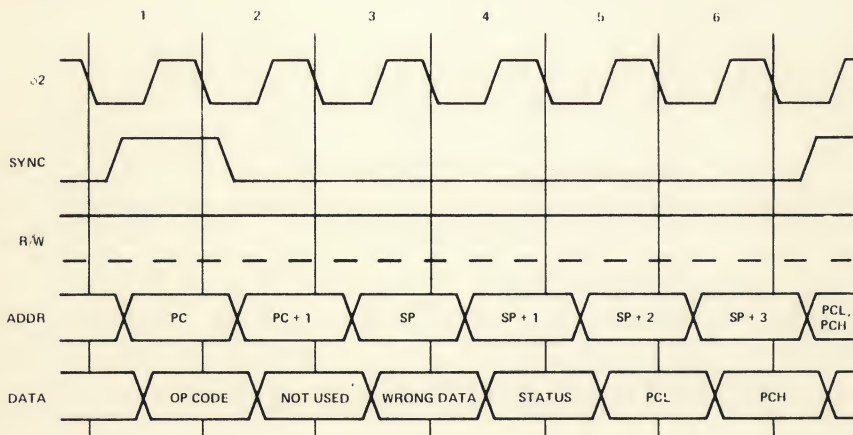
(20) Absolute (JSR)



(21) Implied (RTS)



(22) Implied (BRK)



(23) Implied (RTI)

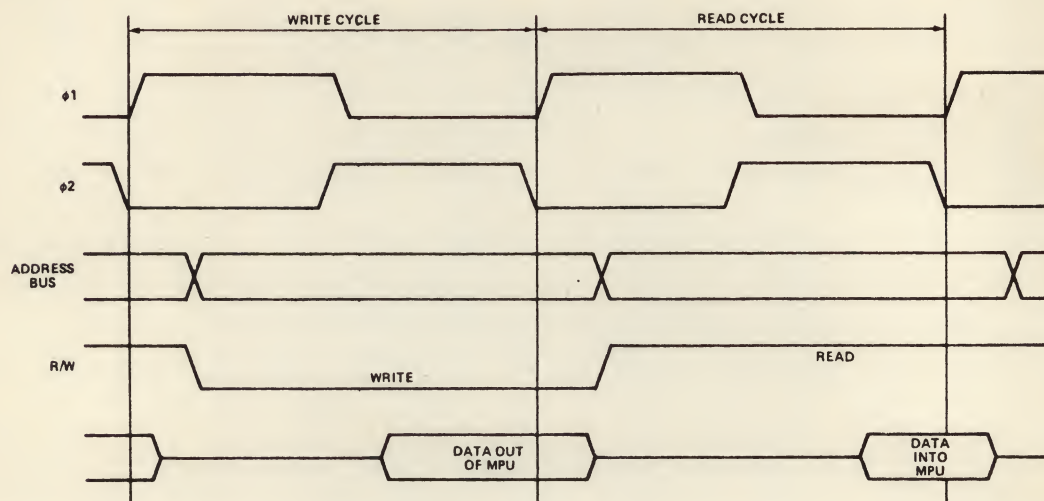


Figure 5.1

Bus Timings

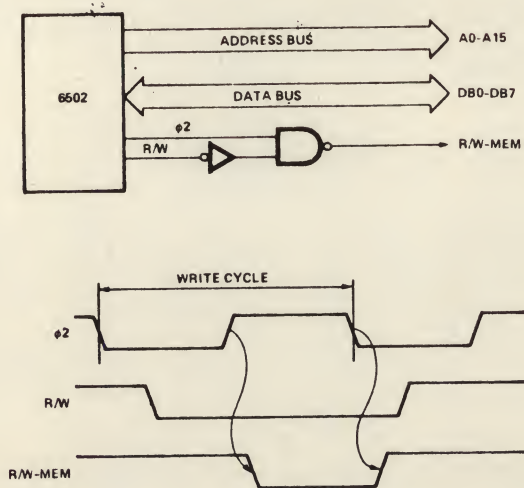


Figure 5.2 Memory R/W Timing

HOOFDSTUK 11 HARDWARE

1. Synchronisatie en timing

Het grootste voordeel van het APPLE-II Personal Computer Systeem is een betere benutting van de TIJD.

Waar anders door ingewikkelde handelingen tijd wordt verkwest , zorgt de uiterst vertijnde en snelle opeenvolging van gebeurtenissen in de APPLE-II ervoor , dat meer tijd beter benut kan worden !

Dit vraagt natuurlijk om een tijdregeling.

Onder de grote bovenklep van de APPLE-II vind je het zgn. moederbord van de computer , waarop naast de elektronica componenten voor procesbesturing en in- en uitlezing van data ook de tijdregeling is ondergebracht.

Wat natuurlijk het meest opvalt op het moederbord is de tamelijk grote microprocessor-chip (6502) , die later uitgebreid aan de orde zal worden gesteld. De microprocessor is middels een aantal meeraderige kabels met de andere delen van het systeem verbonden. Zo ' n meeraderige kabel wordt ' bus ' genoemd. In het hierna volgende schema zien we de ' bus - structuur ' van de APPLE-II. Je ziet , dat de tijdregeling van het systeem in een afzonderlijk blok is ondergebracht. Dit blok , het ' reference oscillator and system timing ' - blok , zorgt voor een juist verloop van de gebeurtenissen in de microprocessor en andere noodzakelijke onderdelen. Zo levert het blok een 1 MHz signaal voor de microprocessor, een 14 MHz en een 7 MHz signaal voor de videogenerator. De oscillator ontleent zijn basisfrequentie van een kristal.

De frequentie van het kristal bedraagt 14.318 MHz.

Vervolgens is het blok ' Sync counter ' van belang. Dit levert signalen , die nodig zijn voor het 'opfrissen' (= refreshen) van het willekeurig toegankelijk lees - schrijfg geheugen (RAM) en het beeldscherm. Het opfrissen is nodig , omdat het geheugen bestaat uit dynamische componenten.

Je kunt je dit voorstellen als een soort van lek in de lading van een geheugenlokatie. Hierdoor zou op den duur informatie verloren gaan. Om dat nu te voorkomen , wordt er regelmatig wat lading toegevoerd , waarmee de lokatie wordt opgefrist.

Hetzelfde geldt voor het beeldscherm. De nawerking van het beeldscherm , maar ook van ons oog , is beperkt. Daarom dient het beeldscherm regelmatig opnieuw beschreven te worden.

In het geheugen komen dan ook speciale gebieden voor waar de beeldscherm informatie wordt verwerkt en opgefrist.

Je zult die gebieden later in de beschrijving van de geheugenmap van de APPLE-II tegenkomen.

De videogenerator levert signalen , die gebruikt worden voor de besturing van de monitor - televisie. Daartoe behoren signalen voor de lijn - en beeldsynchronisatie , de kleurinformatie en intensiteit.

Het ROM - blok spreekt natuurlijk voor zichzelf. ROM betekent ' read only memory ' : exclusief uitleesgeheugen.

De omvang van dit blok is maximaal 6 x 2 Kilobyte. De bijbehorende adressen zijn D000 t/m FFFF. De ROM ' s bevatten de APPLE-II systeem software , zoals APPLESOFT.

Het willekeurig toegankelijk lees - schrijf geheugen van de APPLE-II kan worden uitgevoerd met 4K of 16K dynamische RAM's. RAM betekent 'random access memory'. Als 4K chips worden toegepast , zal een bepaald adres op een andere chip aanwezig kunnen zijn , dan bij een uitvoering met 16K chips.

Om ervoor te zorgen , dat bij een bepaald adres de juiste chip wordt geselecteerd , is het blok 4K/16K RAM Select aanwezig.

Een onderdeel van dit blok vormen de jumperplugjes , waarmee aangegeven wordt , welke geheugenchips in je systeem zijn aangebracht.

Het refreshen van het RAM en de videobeeldbuis gebeurt met een betrekkelijk hoog tempo. De tijd tussen twee 'opfris' - cycli bedraagt hoogstens enkele milliseconden (RAM).

De RAM Adress Multiplexer zorgt ervoor , dat het RAM wordt geselecteerd door de processor , of de ' direct memory access ' controller (DMA controller) , danwel door de Sync Counter voor een RAM of video refresh. Het blok On-board I / O bevat de adresdecoder voor de verschillende selectiesignalen , de A / D omzetters voor de potmeters voor spelletjes (de zgn. game - paddles) , de cassette in - en uitgang en de versterker voor de luidspreker. De A / D omzetters zetten analoge signalen om in digitale.

Het Periferal I / O bevat de selectielogica voor de zgn. ' slots ' (de langwerpige ' stopkontakten ') en de slots zelf.

Hierna volgen enkele schema ' s , die op het APPLE-II Personal Computer Systeem van toepassing zijn.

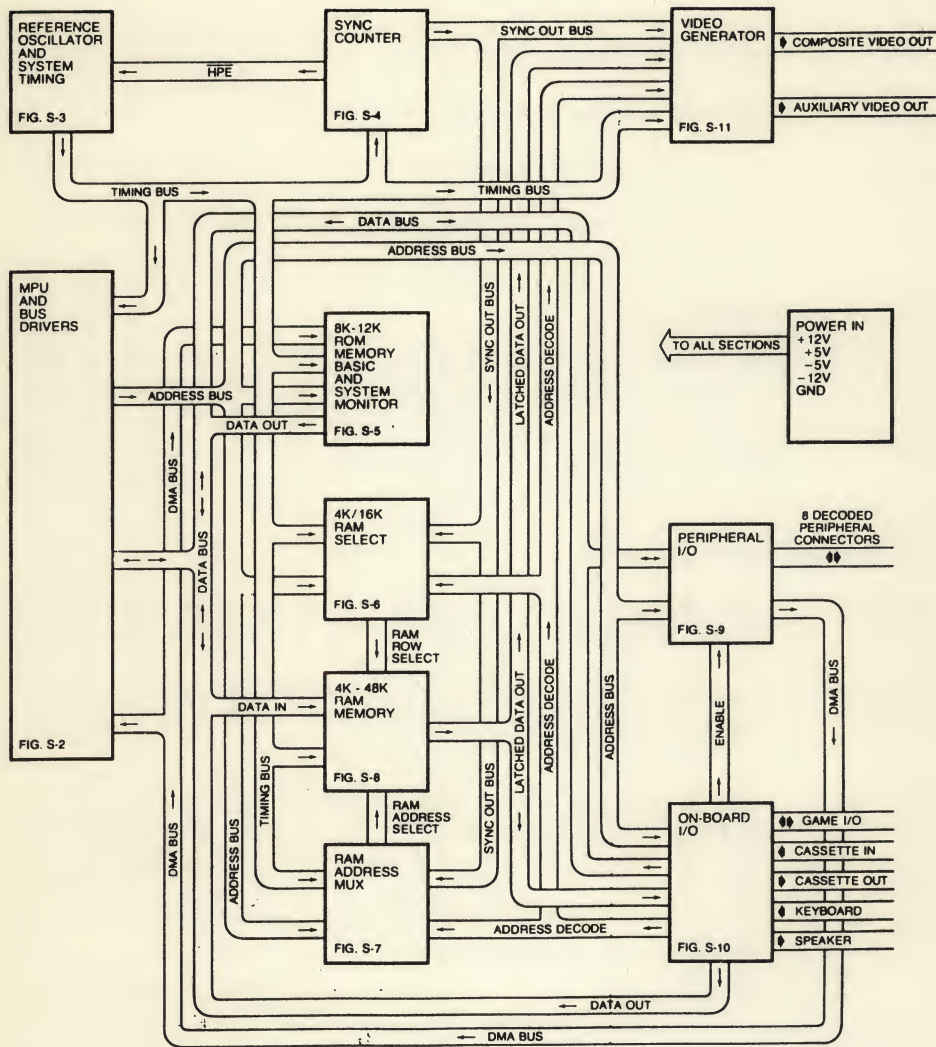


FIGURE S-1 APPLE II SYSTEM DIAGRAM

SYSTEM TIMING

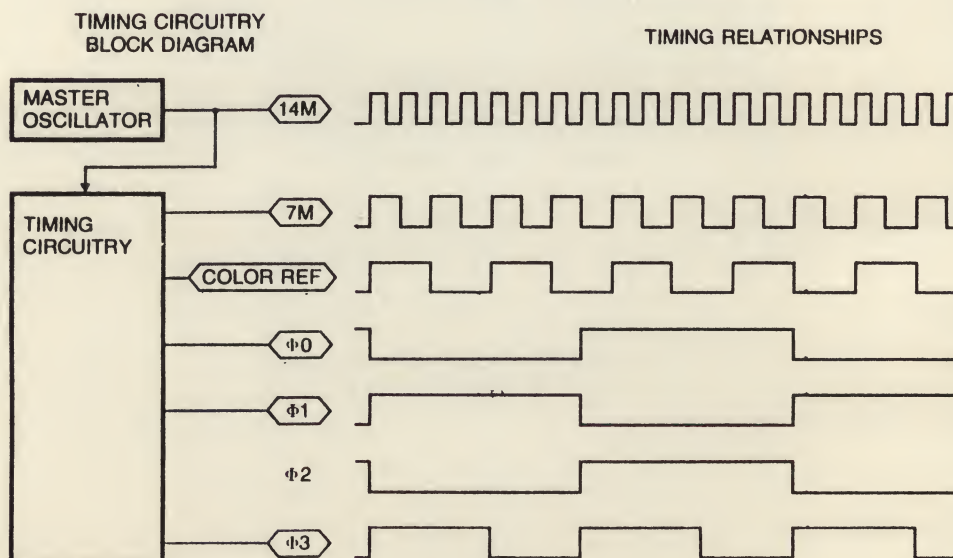
SIGNAL DESCRIPTIONS

- 14M: Master oscillator output, 14.318 MHz +/- 35 ppm. All other timing signals are derived from this one.
- 7M: Intermediate timing signal, 7.159 MHz.
- COLOR REF: Color reference frequency used by video circuitry, 3.580 MHz.
- ϕ_0 : Phase ϕ clock to microprocessor, 1.023 MHz nominal.
- ϕ_1 : Microprocessor phase 1 clock, complement of ϕ_0 , 1.023 MHz nominal.
- ϕ_2 : Same as ϕ_0 . Included here because the 6502 hardware and programming manuals use the designation ϕ_2 instead of ϕ_0 .
- Q3: A general purpose timing signal which occurs at the same rate as the microprocessor clocks but is nonsymmetrical.

MICROPROCESSOR OPERATIONS

- ADDRESS: The address from the microprocessor changes during ϕ_1 , and is stable about 300nS after the start of ϕ_1 .
- DATA WRITE: During a write cycle, data from the microprocessor appears on the data bus during ϕ_2 , and is stable about 300nS after the start of ϕ_2 .
- DATA READ: During a read cycle, the microprocessor will expect data to appear on the data bus no less than 100nS prior to the end of ϕ_2 .

SYSTEM TIMING DIAGRAM



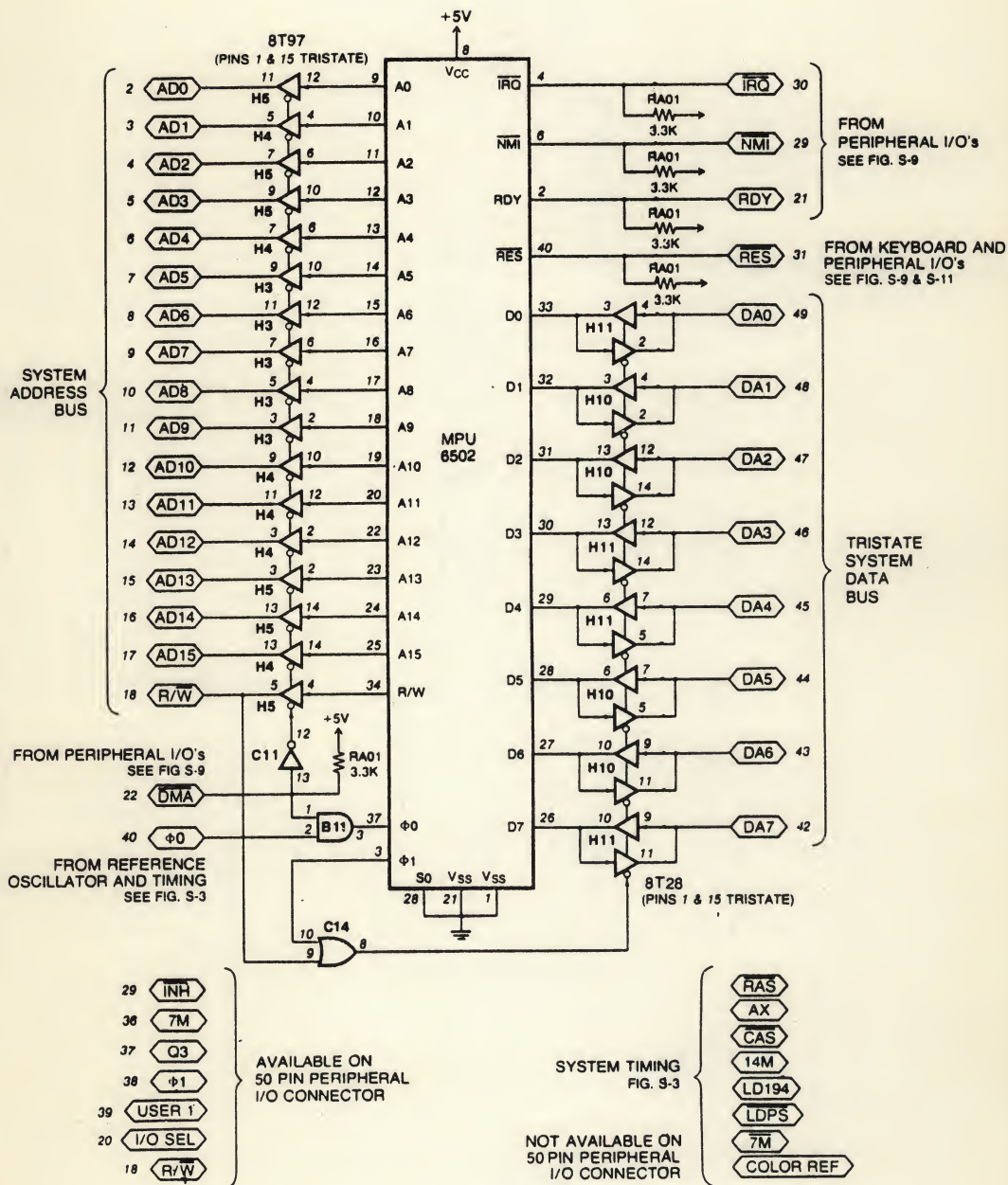
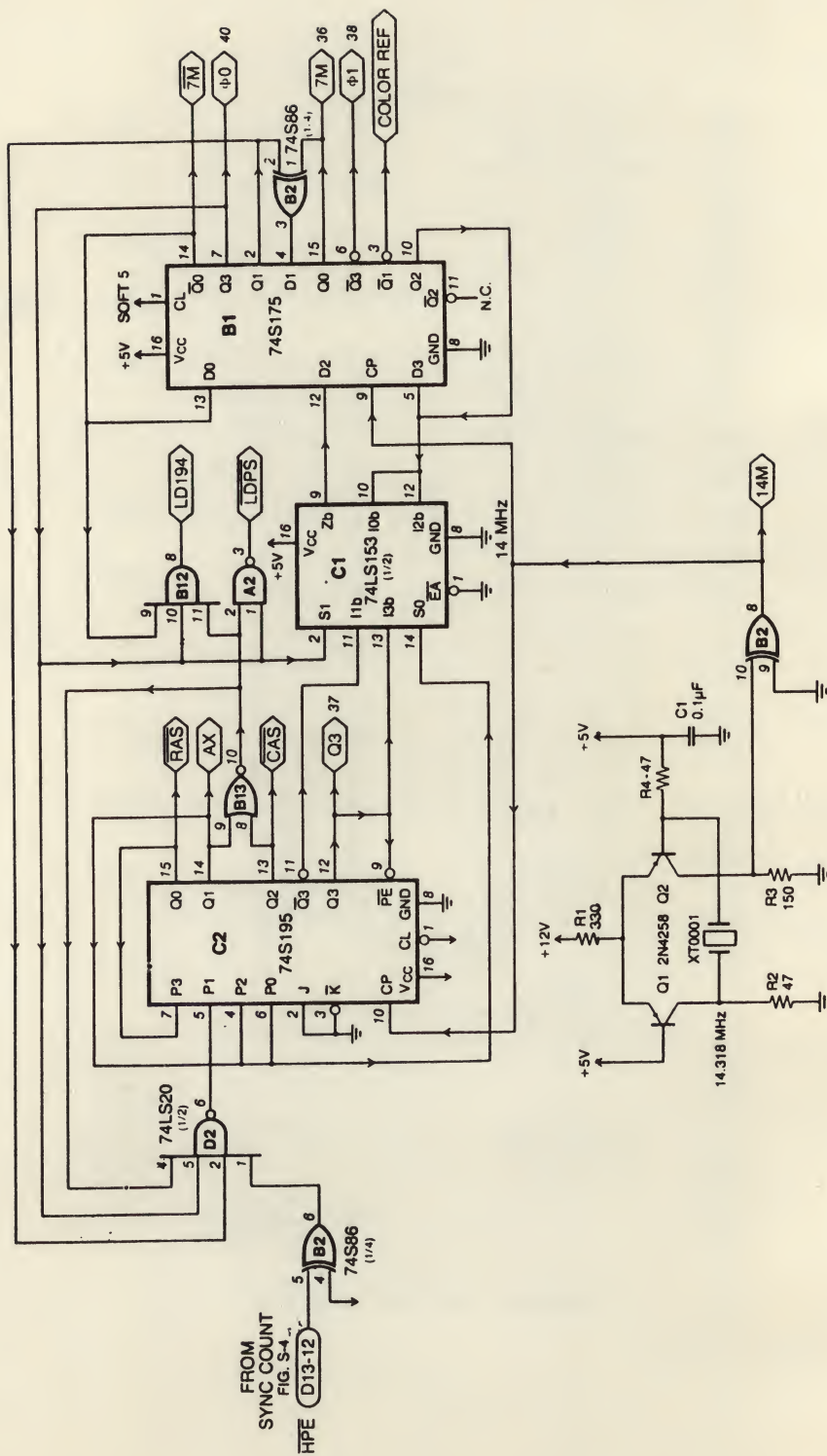


FIGURE S-2 MPU AND SYSTEM BUS



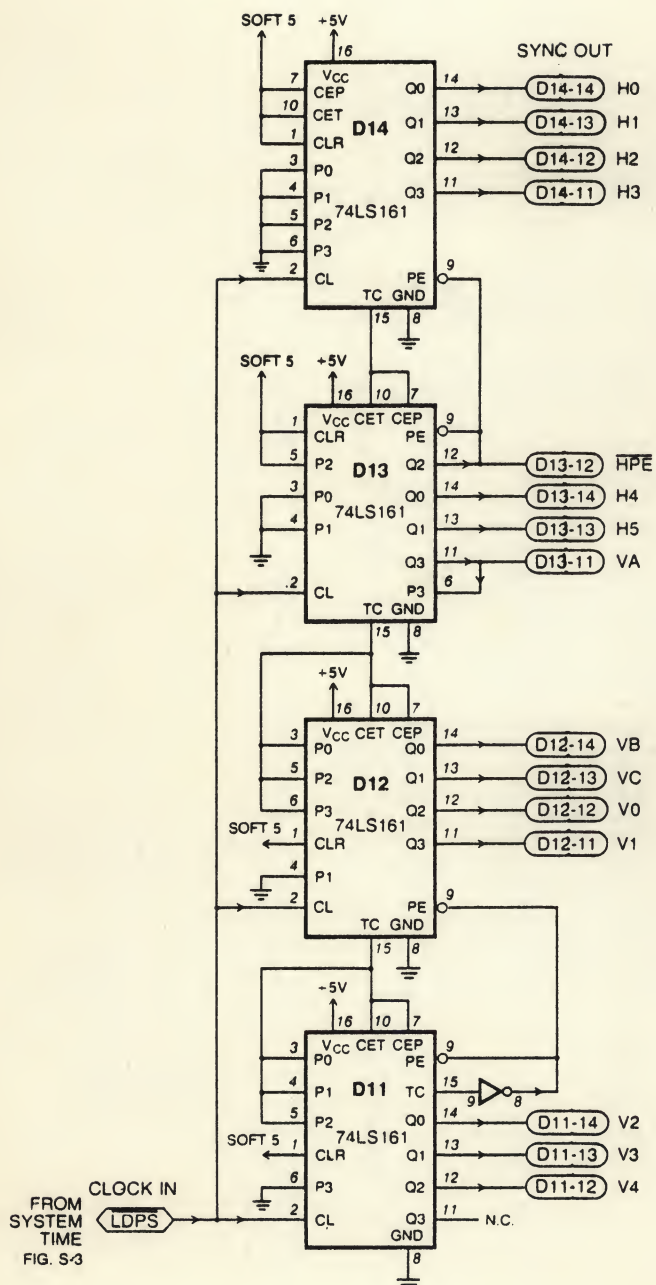
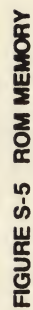
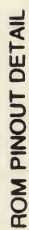


FIGURE S-4 SYNC COUNTER



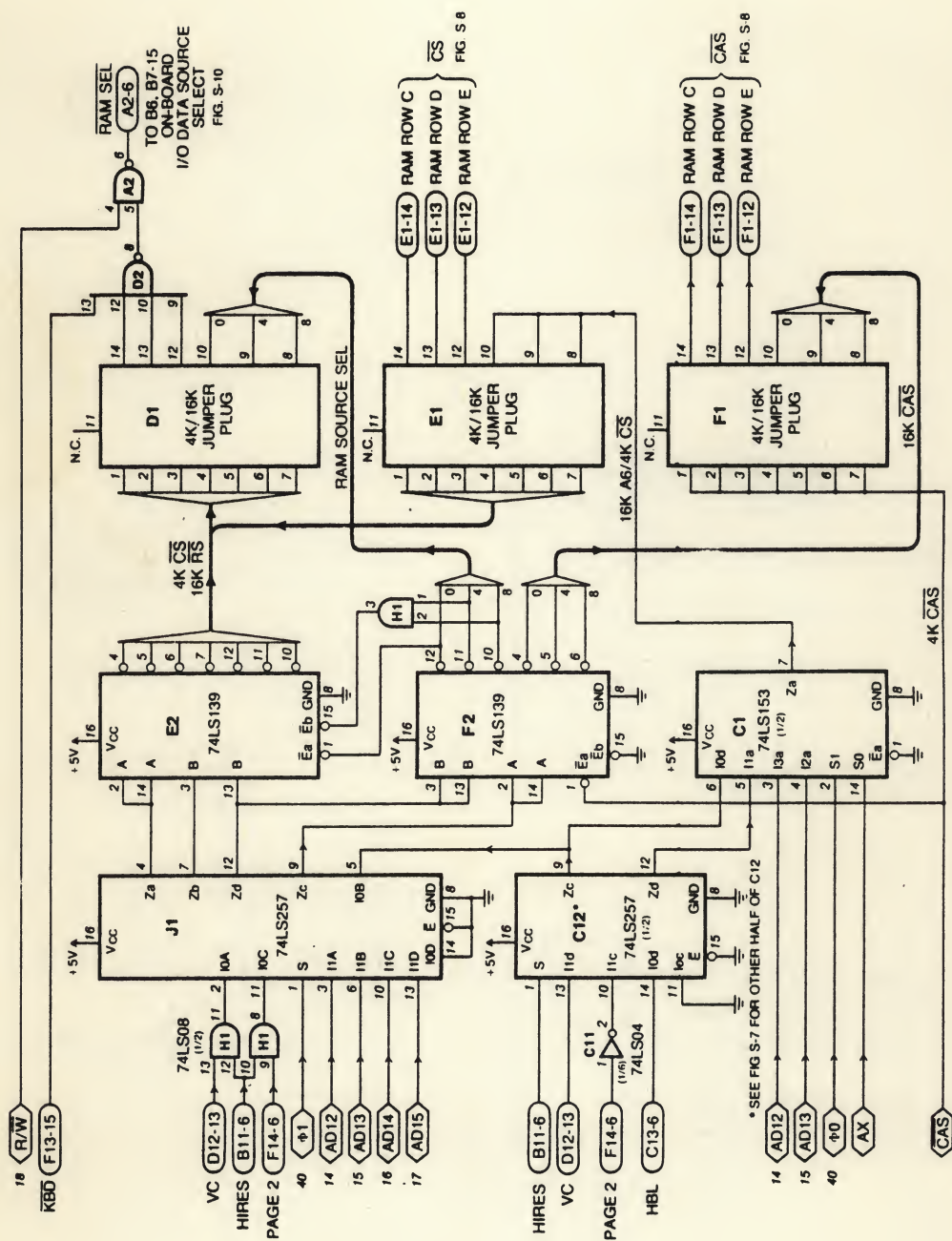


FIGURE S-6 4K/16K RAM SELECT

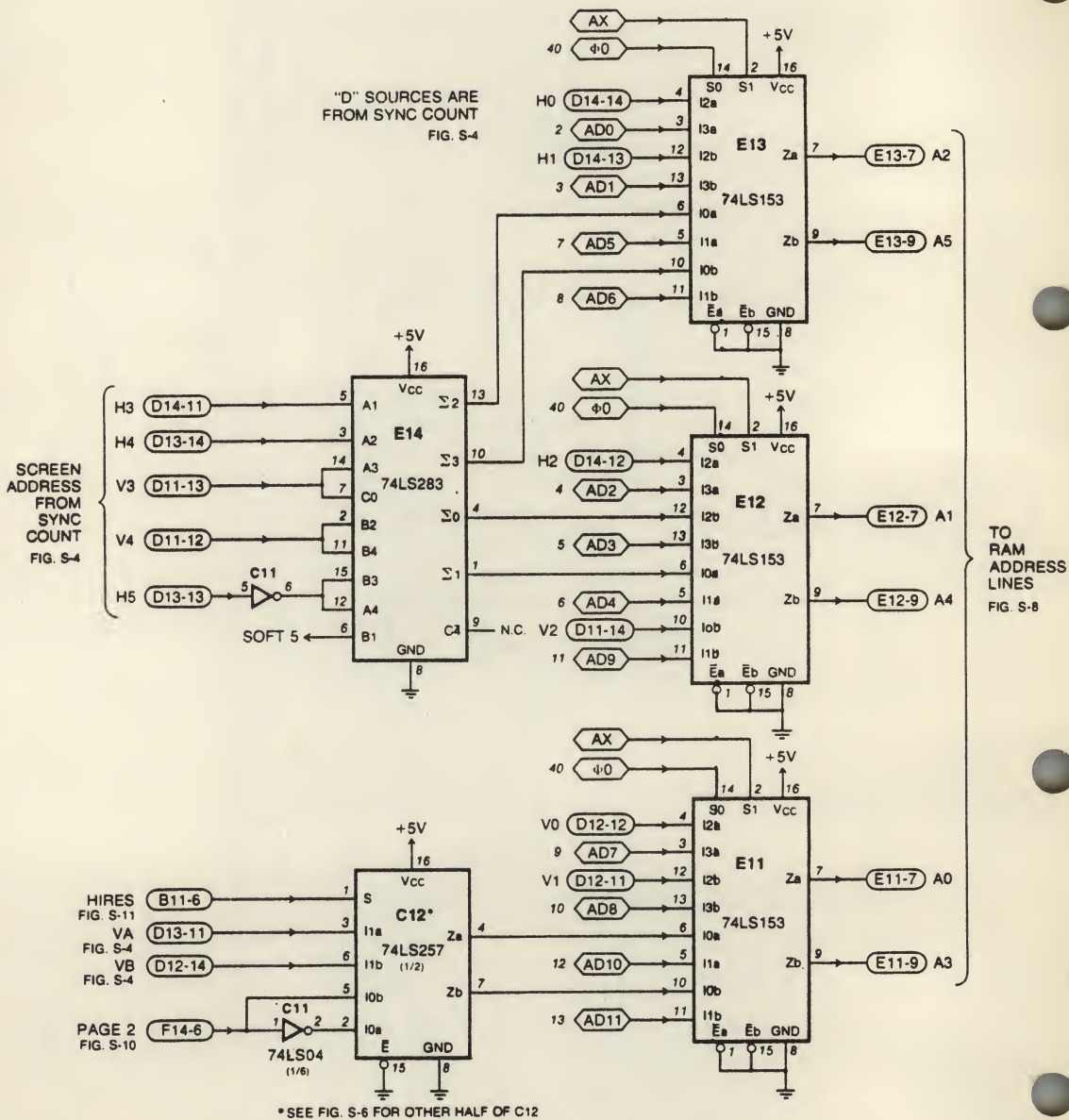


FIGURE S-7 RAM ADDRESS MUX

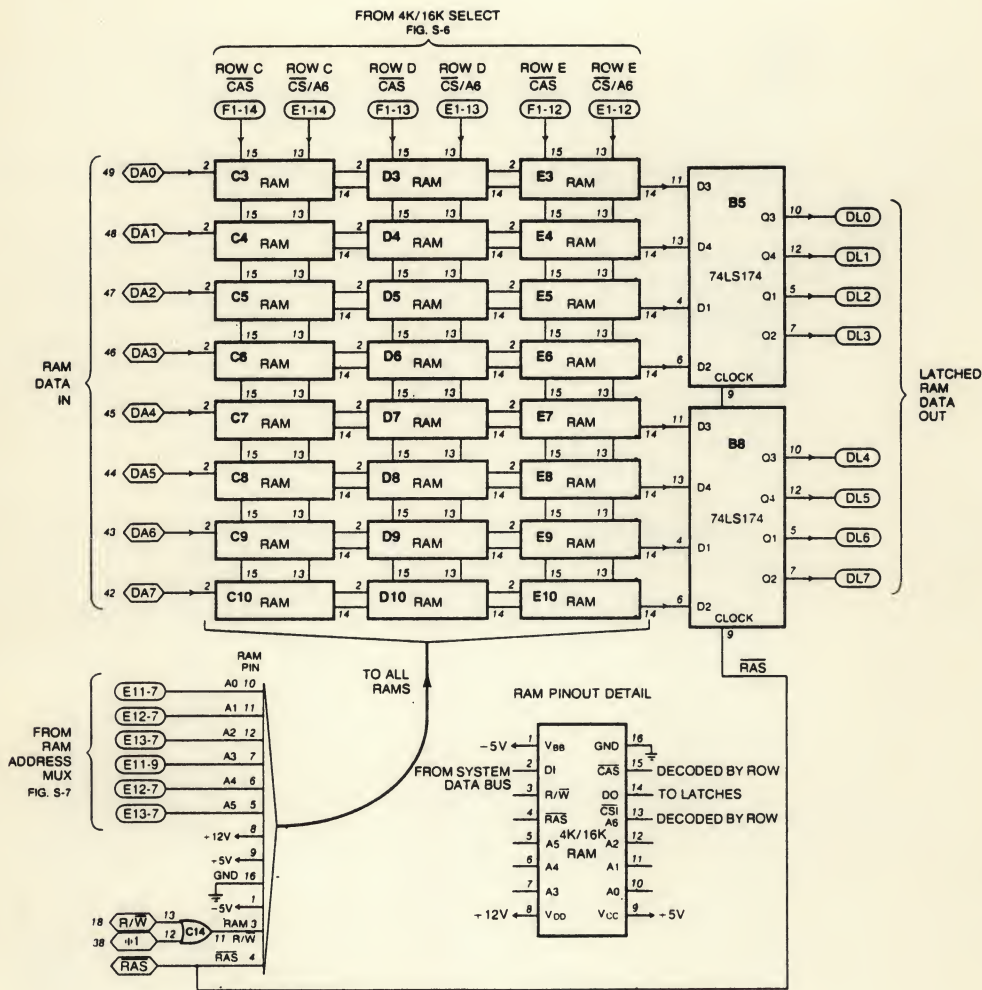


FIGURE S-8 4K TO 48K RAM MEMORY WITH DATA LATCH

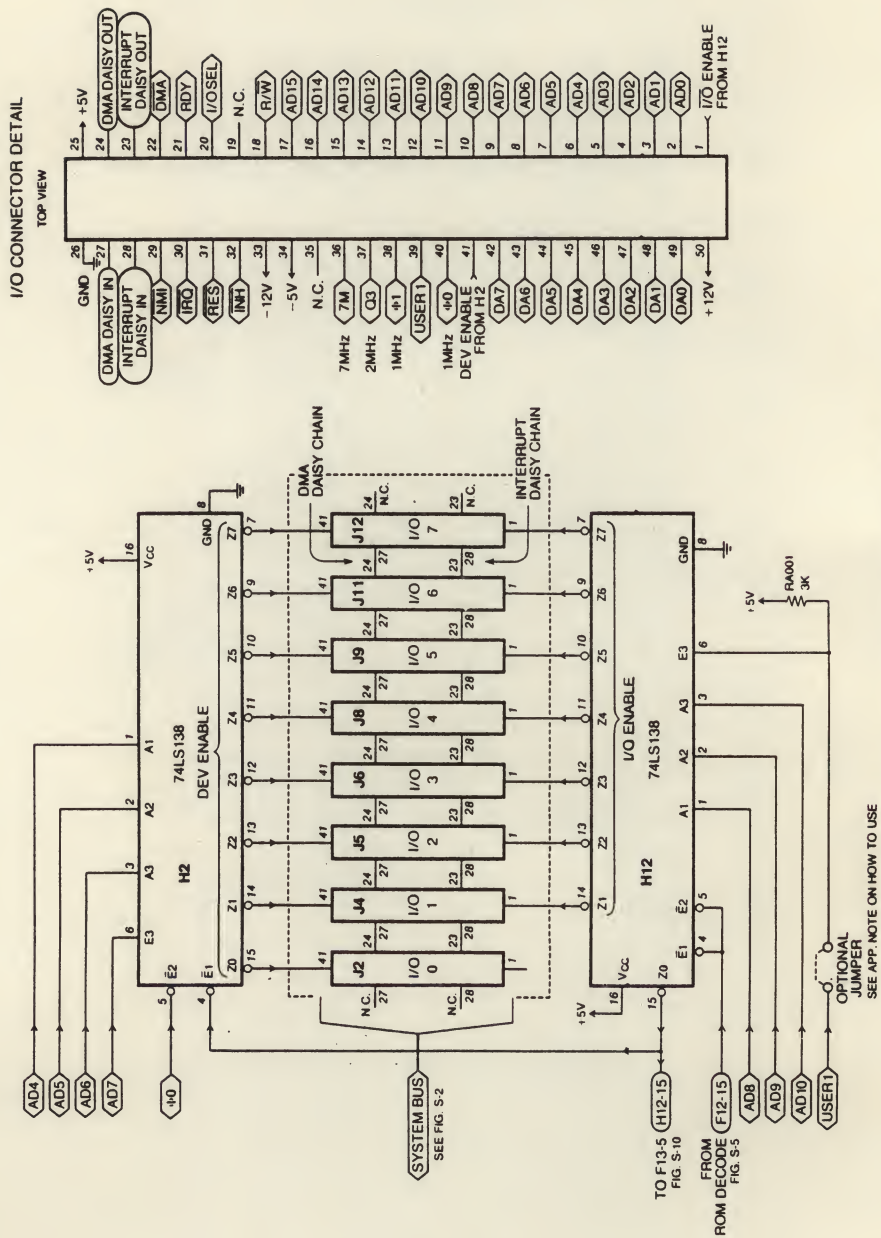


FIGURE S-9 PERIPHERAL I/O CONNECTOR PINOUT AND CONTROL LOGIC

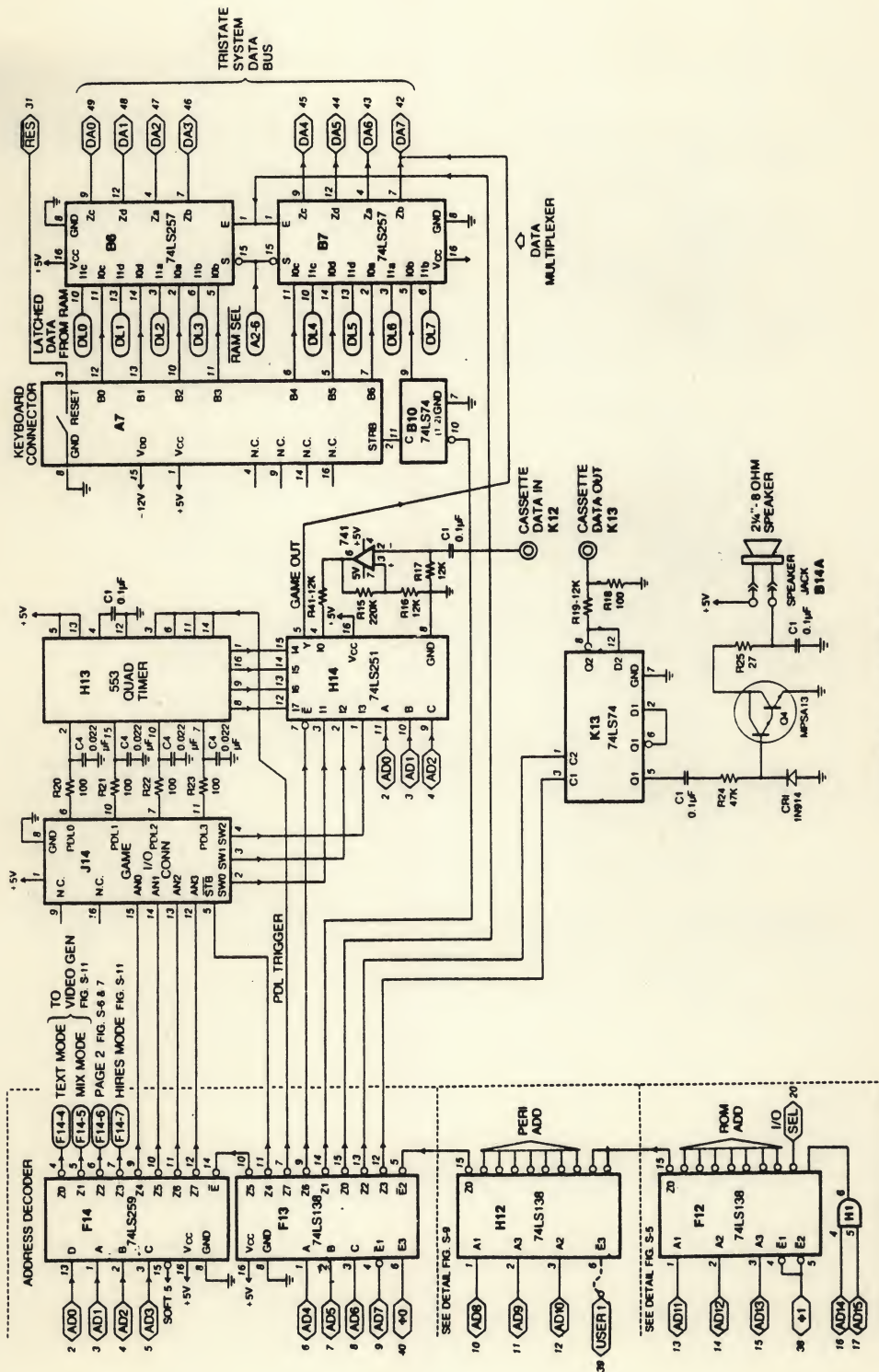
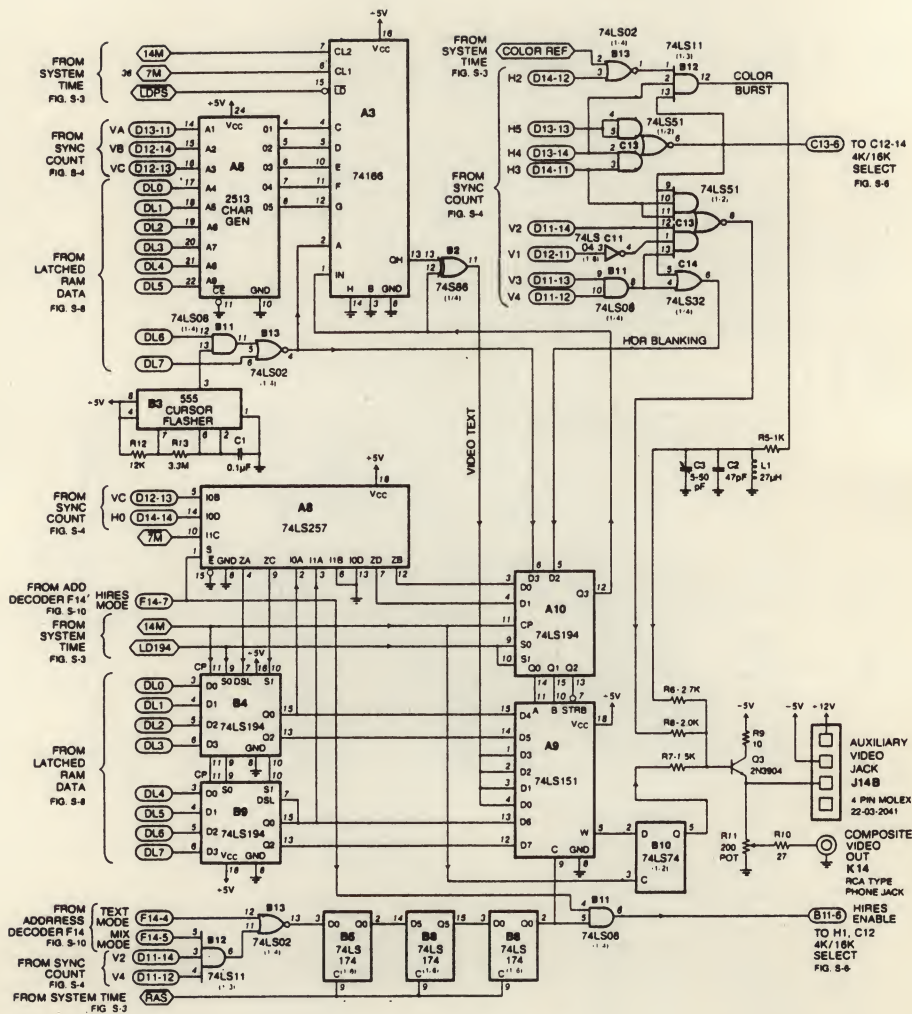


FIGURE S-10 ON-BOARD I/O



2. ROM en PROM beknopt

De APPLE-II heeft een adresbereik van 64K (= 65536 adressen). De bovenste 12K zijn gereserveerd voor ROM/PROM/EPROM. Er zijn op het bord 6 sockets aanwezig voor chips van 2K x 8 bit.

In de geheugenmap is aangegeven , hoe de systeem - software over het ROM - block is verdeeld.

3. RAM

De APPLE-II is ontworpen om te worden gebruikt met dynamisch RAM , opgebouwd uit chips , type 4116, die 16384 (16K) woorden van 1 bit bevatten. Omdat de APPLE-II gebouwd is rond een 8 bits microprocessor , moeten we steeds 8 van deze chips parallel zetten om woorden van 8 bits te krijgen.

Er kunnen maximaal 3 rijen van 8 RAM chips in het moederbord worden gestoken.

4. De geheugenmap

De 64K geheugencapaciteit van de APPLE-II is op de volgende wijze toegedeeld :

0000 - BFFF	(48K)	RAM
C000 - CFFF	(4K)	I/O
D000 - FFFF	(12K)	ROM

In de volgende tabellen is aangegeven , waarvoor de diverse gebieden worden gebruikt.

De eerste tabel geeft een overzicht in blokken van 4K.

De tweede tabel geeft meer gedetailleerd weer , hoe de eerste 3K van het geheugen worden gebruikt.

De derde tabel tenslotte laat zien , hoe het gebied , dat bestemd is voor I / O en speciale functies is ingedeeld.

Memory Address Allocations in 4K Bytes

0000	text and color graphics display pages, 6502 stack, pointers, etc.	8000	
1000		9000	
2000	high res graphics display primary page	A000	
3000	"	B000	
	"		
4000	high res. graphics display secondary page	C000	addresses dedicated to hardware functions
	"		"
5000	"	D000	ROM socket D0: spare
	"		"
6000	"	E000	ROM socket E0: BASIC
	"		"
7000	"	F000	ROM socket F0: BASIC utility
			ROM socket F8: monitor

Memory Map Pages 0 to BFF

HEX ADDRESS(ES)	USED BY	USED FOR	COMMENTS
PAGE ZERO 0000-001F	UTILITY	register area for "sweet 16" 16 bit firmware processor.	
0020-004D	MONITOR		
004E-004F	MONITOR	holds a 16 bit number that is randomized with each key entry.	
0050-0055	UTILITY	integer multiply and divide work space.	
0055-00FF	BASIC		
00F0- 00FF	UTILITY	floating point work space.	
PAGE ONE 0100-01FF	6502	subroutine return stack.	
PAGE TWO 0200-02FF		character input buffer.	
PAGE THREE 03F8	MONITOR	Y _c (control Y) will cause a CJSR to this location.	
03FB		NMI's are vectored to this location.	
03FE-03FF		IRQ's are vectored to the address pointed to by these locations.	
0400-07FF	DISPLAY	text or color graphics primary page.	
0800-0BFF	DISPLAY	text or color graphics secondary page.	BASIC initializes LOMEM to location 0800.

HOOFDSTUK 12 TROUBLE SHOOTING

1. Enkele oorzaken

Hoewel de APPLE een bijzondere robuuste bouw heeft en de onderdelen van uitstekende kwaliteit zijn, kan het gebeuren, dat er een storing optreedt. De programma-executie stopt, er verschijnen vreemde tekens in het beeld, of de monitor valt weg, de diskdrives doen raar, en je printer weigert; allemaal redenen, om je ontzettend bezorgd over te maken. Toch zijn er een (klein) aantal "standaard-oorzaken" aan te geven, die klachten veroorzaken. Klachten, die veelal met eenvoudige middelen zijn te achterhalen en te verhelpen. Een van die klachten betreft het al meermalen gesignaleerde "stilstaande lucht" syndroom. De computer dient opgesteld te staan in een goed ventilerende ruimte.

Helaas zal het merendeel van de APPLE-gebruikers naar een service- centrum van APPLE Computers moeten gaan, wanneer zich kleine storingen voordoen. Misschien dat enkele aanwijzingen voldoende zijn, om niet voor ieder wissewasje naar uw leverancier te gaan.

Enkele standaard-"klachten"

De aan/uit schakelaar geeft een "krakend" geluid bij het schakelen. Remedie is er eigenlijk niet voor. Bij het schakelen kunnen er vonkjes tussen de kontakten overspringen, waardoor het eigenaardige geluid ontstaat. Schakel wat resoluter.

De monitor-TV doet het niet, of geeft een flauw, of een gestoord beeld. Controleer de kabel van de TV-ingang van je monitor en let vooral op de gesoldeerde (?) plug. Beweeg de draad - wat doet het beeld. Dit is meestal geen computerdefect.

Het "Power"-lampje gaat niet aan. Doet de computer het wel? Oplossing is het lampje te vervangen. Zie Reference Manual.

Is de computer wel op netspanning met randaarde aangesloten?

De diskdrive draait, maar leest de diskette niet uit. Hij "boot" niet. Gebruik je een 13 of 16 sectoren diskette (DOS 3.2/3.3)?

Gebruik een masterdiskette en controleer alles opnieuw. Indien dit geen uitkomst brengt, dan wordt de computer uitgezet. Zet de stroom af, maar houdt de leiding in de voeding (aarding, weet je wel!). Is het "Power"-lampje uit ? Ja, dan klep van de computer open doen en interfacekaart lokaliseren. Licht hem voorzichtig, maar vastberaden uit het Slot. Is hij stoffig. Controleer de kontakttongen. Steek hem ferm terug op zijn plaats. Deksel nu weer dicht. Power aan en opnieuw proberen. Meestal oxidatie van de interfaceconnectoren. Even "jutteren" is vaak voldoende. Maar weest voorzichtig! Zeer zelden zijn het de kleurrijke "flatcables" van de drives. Minder zelden is het helaas de besturingslogica op het analog board in de drives. Vermijdt statische electriciteit. Houdt bij het manipuleren met een hand de APPLE voeding vast. Is het analog board defekt, informeer dan bij verschillende APPLE leveranciers naar de mogelijke kosten.

De computer "hangt" telkens en weigert normaal te werken. Is dit spontaan gekomen, of na aanraking van chips in het binnenste van de computer ? Is het een spontaan probleem, dan moet je APPLE terug naar de dealer. Statische electriciteit (kleding, vloerbedekking) is vaak een oorzaak na het bekijken en betasten van de chips. Via de leverancier of hobbyclub zijn testprogramma's te betrekken, die de defekte chip lokaliseren. Kosten zijn meestal minimaal. Zie verder onder "vervangen van chips".

Toets dendert. DDDus, hij geeft bij het indrukken meer dan 1 karakter. Schoonmaken is vaak geen oplossing. Beter is vervanging van de defekte toets en controle van de keyboard encoder, liefst via een goed geoutilleerde leverancier. DDDenderen uw vingers soms ook wel eens....??? De goedkoopste remedie hiervoor is harder tikken!

Na uitgebreid met de diskdrives gewerkt te hebben, schijnt de computer niet goed meer te gehoorzamen. Lees- en of schrijffouten worden gemeld. Mogelijk is het "vollopen" van het geheugen oorzaak. In computerbargoens heet dit fenomeen "clobbering". Het beste is, de computer enkele momenten af te zetten. Theoretisch is het voldoende het commando: X=FRE(0) en/of FP te gebruiken, om het systeem te "schonen".

Uitgebreide moeilijkheden met cassette-opnamen worden behandeld onder het hoofdstuk Interfacing. In 99 van de 100 gevallen is de recorder met zijn aansluitingen de oorzaak.

Heeft u bij het uitpakken van uw APPLE de meegeleverde handboeken ook werkelijk bestudeerd ?

2. Vervangen van IC's

De meeste geïntegreerde circuits (IC's) van de APPLE zijn standaard chips, die tegen uitzonderlijk lage prijzen zijn te verkrijgen. De ROM's van het moederbord, de keyboard encoder en analog boards zijn de duurste onderdelen. Het is bij de voorbereiding van dit Handboek opgevallen, dat veel leveranciers niet uitgerust zijn voor adequaat servicen van de APPLE en dat dit blijkt uit het niet aanwezig zijn van de noodzakelijke meetapparatuur, waaronder een oscilloscoop. Ook is het meermalen opgevallen, dat serviceverleners niet met moderne meetapparatuur konden omgaan en dat elke theoretische kennis van de typen chips ontbrak. Hieraan valt nog zeer veel te verbeteren! Reden te meer de eigen capaciteiten na te gaan en af te wegen, of een kleine reparatie niet zelfstandig kan worden uitgevoerd.

Wanneer aan de hand van het schema is vastgesteld welk gedeelte van de computer een storing veroorzaakt, zal afgewogen kunnen worden of vervanging van IC's zelfstandig kan worden uitgevoerd. Je moet de volgende procedure goed in het oog houden.

- a. Schakel de APPLE uit, maar houdt de netleiding in de voeding.
- b. Verwijder het deksel.
- c. Verwijder zonodig de interfacekaarten.
- d. Houdt met een hand af en toe de metalen voeding vast (aarde).
- e. Lokaliseer de defekte chip en verwijder hem met een IC trekker.
- f. Let op de witte, pin 1 wijzer op het moederbord en de dot-markering op de te vervangen chip.
- g. Breng de interfaces weer aan en sluit het deksel.

Moet om wat voor reden ook de kast worden verwijderd, zorg dan, dat de APPLE op de kop wordt neergelegd bij voorkeur op een zachte, maar anti-statische onderlegger. Verwijder de zes kruiskopschroeven in de buitenhoeken van het ondervlak. Verwijder tevens de vier schroeven onder het keyboard.

Hierna kan de APPLE uit de kast worden gehaald. Verbreek nu de verbinding tussen moederbord en keyboard: connector lostrekken.

Wil je het keyboard verwijderen, maak dan de vier schroeven los in de hoekpunten van het keyboard.

Wil je -zonder het uiteen nemen van de APPLE- een toets van binnen schoonmaken, licht dan met een kleine schroevendraaier het toets-letterkapje. Voorzichtigheid is hierbij geboden. Beter is een klein punttangetje, waarvan de bekjes voorzien worden van een stukje plastic (van een boterhammenzakje o.i.d.) voor een betere grip. De toetskapjes zitten rechtstandig in een schuifvatting gedrukt en kunnen daardoor recht omhoog worden getild. Dit geldt ook voor het "power"-lampje.

Het openen van de diskdrives dient uiterst voorzichtig te geschieden. Verwijder de schroeven in het grondvlak en schuif de kast naar achter weg. Let op de regenboogkabel! Let erop, dat te nimmer het analog board bovenop de diskdrive door de kast wordt geraakt en kortgesloten!! Het bijstellen van de DSpeed met de kleine draaipotmeter op het board kan alleen gebeuren, wanneer je beschikt over een testdiskette. Het schoonmaken van de fijne kop van de diskdrive moet voorzichtig gebeuren en gebruik eventueel een cassette reinigingsmiddel. Gebruik GEEN schoonmaak diskette - die bekrassen de opnamekop!!! Schoonmaken van de kop is slechts onder uitzonderlijke omstandigheden nodig. Bekijk gelijk het aandrukviltje van de diskettehouder. Men zegt, dat een heel klein druppeltje naaimachine-olie op de kopgeleistangen geen kwaad kan. Is er een hardwarefout opgetreden in de diskdrive, bijv. leest de drive niet meer, dan hoeft dit niet te betekenen, dat een chip defect is. Nogal vaak blijkt een weerstand en een gelijkrichter het begeven te hebben. Meetapparatuur is dan nodig.

Bij het vervangen van een chip moet je er goed op letten, dat de minuscule kontaktpennen ("pootjes") op hun plaats in de socket terecht komen. Zit een pennetje scheef, of breekt het af, dan zit je in de moeilijkheden. Voorzichtig bijbuigen kan bijvoorbeeld op een vlakke tafelrand. Let erop, dat de IC niet met statische electriciteit in aanraking komt. Gebruik een IC-trekker, liever geen tang. Een postzegelpincet is eventueel ook geschikt. Weet je niet welke IC een storing kan veroorzaken, praat er dan eens over met een APPLE gebruiker in je omgeving. Ben je al lid van de HCC? Neem het dan op met clubvrienden.

BIJ ELKE IWIJFEL AAN DE EIGEN KUNDIGHEID DIEN JE
ZONDERMEER JE LEVERANCIER IN TE SCHAKELN.

HOOFDSTUK 13 ALGEMENE SLOTOPMERKINGEN

1. De Hobby Computer Club

De HOBBY COMPUTER CLUB is een vereniging voor mensen die zich uit liefhebberij bezig houden met computers. Doel ervan is, het stimuleren van kontakten en het informeren van de leden over ontwikkelingen op het gebied van computers. Hierdoor kan men met meer kennis van zaken (en dus genoeg) zijn hobby bedrijven.

Hoewel de HCC in principe ALLE computersliefhebbers wil verenigen, zijn de meeste leden gebruikers van grote en kleine systemen, maar hoofdzakelijk georiënteerd op microcomputers. De snelle groei van de HCC duidt erop, dat zij in een behoefte voorziet. Zowel in België als Nederland is de HCC actief. Ze organiseert bijeenkomsten, lezingen, gebruikersactiviteiten en informeert haar leden ondermeer middels een fraai, dik tijdschrift, de "HCC Nieuwsbrief". Het verschijnt 12 maal per jaar op A4 formaat.

De HCC kent afzonderlijke "gebruikersgroepen", zoals de AGG, de APPLE GEBRUIKERS GROEP. Voorts kent de HCC een hardware afdeling.

Hierbij kunnen de leden artikelen bestellen tegen vaak sterk gereduceerde prijzen. Er is een software afdeling, die op cassettes en diskettes nuttige programmaatjes verspreidt. En de HCC beschikt over een bescheiden uitgeverij, die inmiddels een 16-tal publikaties het licht liet zien. De HCC organiseert jaarlijks een groot verbindend evenement, waarop alle leden en alle leveranciers welkom zijn voor een uitwisseling van informatie. Het is dus geen club voor "salon-computeristen" en aanverwante "societeits-gezelligheids liefhebbers", maar een werkelijk actieve vereniging met een grote dadendrang.

Is het zinvol, om lid te worden van de HCC. Wij geloven van wel. Echter gebiedt de objectiviteit te stellen, dat dit oordeel zeer persoonlijk is. De HCC bevindt zich in het spanningsveld "computerverkoper-koper", waarin tegengestelde belangen ten volle voor het voetlicht kunnen treden. De tamelijk "kille" houding van sommige leveranciers t.a.v. de HCC en de pressie van de HCC op leveranciers om in te haken op bepaalde HCC-activiteiten (w.o. de wens speciale prijzen te mogen rekenen voor leden), laat de buitenstaander weleens afschrikken. Dat de HCC een "hobby copieer club" is, waarbij men kostbare software voor een habbekrats kan betrekken, is een flagrante onwaarheid! Degenen, die daarom lid worden, komen bedrogen uit. Door de snelle toename van het aantal priveecomputers ligt de groei van de HCC voor de hand. Helaas blijkt een zinvolle communicatie tussen de leden buiten de grote stadscentra -naast de "HCC Nieuwsbrief" en de "Gebruikersgroep"- te wensen over te laten. Computeristen zijn individualisten, ofschoon bewezen kan worden dat "samenwerking individuele versterking" betekent. Hierin vervult de HCC een belangrijke taak.

Conflict Simulatie :

In dit boek hebben we getracht je bekend te maken met de veelzijdigheid van de APPLE-II Personal Computer. Het is je inmiddels wel duidelijk geworden , dat het systeem meer kan dan alleen maar spelletjes.

Maar aangezien computers en spelletjes nauw met elkaar verweven zijn (op de eerste ' echte ' computer , de ENIAC , werd al een variant op het bekende boter , kaas en eieren - spel gespeeld !!!) willen we voor de volledigheid het ' spelelement ' ook nog even aanstippen. De eerste Personal Computers , waaronder ook onze , nu dierbare , APPLE waren in wezen spelletjesmachines. Later bleek pas de veelzijdigheid van de Microprocessor. Dit verklaart ook het enorme arsenaal van spelletjes voor de APPLE. En zeg nu zelf , het is best leuk om een pot STARTREK , STARWARS , GALAXIAN , INVADERS of MIDWAY te spelen.

Echter na verloop van tijd zullen deze spelletjes gaan vervelen , omdat men de structuur van het programma doorkrijgt.

Vandaar ook dat er verbeterde versies of geheel nieuwe versies op de markt zijn gekomen. Deze versies stellen je in staat van te voren het niveau van je spel in te stellen , het aantal tegenstanders , snelheid en zo voort.

Deze soms zeer moeilijke computerprogramma ' s kunnen we dan ook eigenlijk geen spelletjes meer noemen. Een beter woord is ' simulatie ' , d.i. nabootsing

Dat computers geschikt zijn voor het simuleren van de werkelijkheid , zal een ieder bekend zijn. Voorbeelden hiervan zijn de Vluchtnabootsers voor Burgerluchtvaart , Militaire luchtvaart en natuurlijk de Ruimtevaart (je dacht toch niet dat de heren Crippen en Young de Spaceshuttle voor het eerst direkt op een landingsbaan zetten ?) .

Wat je waarschijnlijk niet weet , is dat jouw computer ook tot dit soort simulaties is staat is . De APPLE gebruiker is hierbij bijzonder in het voordeel vanwege de enorm krachtige HIRES mogelijkheid.

Ook op de APPLE zijn vluchtsimulaties mogelijk , een voorbeeld hiervan is de FS = 1. Andere simulaties zijn Air Traffic Controller , Super Star Trek en 3D Docking. Een voor jou geheel onbekend terrein is waarschijnlijk de Computer CONFLICT Simulatie. Hiervoor zijn fantastische programma ' s in de handel. Deze simulaties variëren van de Oudheid (Roman Empire , Computer Napoleonic) via heden (Computer Ambush) tot ver in de toekomst (de , inmiddels vier , delen van de GALACTIC saga) .

Deze uitgekiende programma ' s stellen je in staat om grootscheepse conflicten op jouw computer te simuleren. Een spelduur , variërende van enige uren tot enige weken (jazeke) , is eerder regel dan uitzondering.

Bescheiden voorbeelden van een conflict simulatie zijn de programma ' s RISK en WARLORDS , die respectievelijk voorkomen op de diskettes 2A en 2B van de HCC.

Nu bestaat er in Nederland een Vereniging van mensen die de conflict - simulatie als gemeenschappelijke hobby hebben.

Deze Vereniging genaamd DUCOSIM (DUTch CONflict SIMulation association) is een overkoepelende organisatie voor onderafdelingen , die ieder in een bepaalde interessesfeer werkzaam zijn .

Zo is er de ' Miniature - Wargaming ' , de ' Board - Wargaming ' , ' Fantasy/Science Fiction Wargaming ' en natuurlijk een afdeling ' Computer - Spellen '.

Deze computer-poot van DUCOSIM heet DUCOCOSIM (de eerste CO van COmputer natuurlijk).

DUCOSIM heeft zich als doel gesteld ' het contact tussen alle Nederlandstalige wargamers (miniaturen , bordspellen , fantasy/science fiction en diplomacy) te onderhouden en te versterken en belangstelling te wekken voor onze gemeenschappelijke hobby '.

Men probeert dit te realiseren door :

1. Het uitgeven van een maandblad ' Conflictgazet ' met daarin verenigingsnieuws , aankondigingen van conventies etc.
2. Het organiseren van conventies , waar leden uit het gehele land elkaar treffen om hun hobby te beoefenen of er over van gedachten te wisselen.
3. Het organiseren van kampioenschappen.
4. Het actief steunen van leden , die zich inspannen om lokale of regionale afdelingen van DUCOSIM op te zetten.

Belangstellenden kunnen schrijven naar :

DUCOSIM

Van Foreestweg 38
2614 CK Delft

3. DAAR MAKEN WE EEN KOPIETJE VAN

Je bent net begonnen met je APPLE hobby. Dit boek wil een samengevatte handleiding zijn, die het je mogelijk wil maken snel wegwijs te worden in het gebruik van je geweldige computer. Je zult nu wel inzien, dat de computer een "dom" ding is, dat eerst voorgeprogrammeerd moet worden voor zijn taak. Nu haat je waarschijnlijk net als wij het telkens uitvinden van weer een nieuw "wiel" - dus zoek je naar pasklare programma's, die je zo kunt gebruiken. Programmatuur, of beter: software, is m.n. voor de APPLE II computer in uitzonderlijke hoeveelheden te krijgen. Hoewel de makers van deze software hun inzet graag goed beloond willen zien (hoeveel regels kun jij nu per uur programmeren?), zijn de pakketprijzen erg laag. Een spelletje kost zelden meer dan Hfl.100,-/Bfr.1400,-. Een database krijg je zelfs kostenloos bij aanschaf van een diskdrive; o waar die dan wel te vinden is ? Kijk eens op je masterdiskette

Een redelijke gegevensverwerker kost tussen Hfl.450/Bfr.6300,- en Hfl.2500,-/Bfr.35000,-. Toch zijn dat vaak "onoverkomelijke" sommen, die enkelen er toe bewegen te gaan zoeken naar "gestolen" of "gekopieerde" versies van het oorspronkelijke programma. Het plegen van inbreuk op vermeende auteursrechten op software is daarmee een feit. Dit onderwerp ligt zeer gevoelig - want waar houdt het auteursrecht op en waar begint de diefstal c.q. de inbreuk op dit recht. Hans van Kampen formuleerde onlangs vier "wetten", die een aanzet kunnen zijn tot een zinniger discussie over de plaats van software in het automatiseringsgebeuren. Hij noemde zijn "wetten" de "Final Four":

- 1-Slechte software wordt onkopieerbaar gemaakt
- 2-Goede software is direkt kopieerbaar
- 3-Goede software wordt niet illegaal gedistribueerd
- 4-Er bestaan maar 4 verschillende softwarepakketten

En ieder, die zich ervan bewust is, dat "backup copies" van diskettes noodzakelijk zijn, zal so-wie-so de Wet 1 en 2 onderschrijven. Echter lijkt het moeilijk een programma als "VISICALC" met "slecht" te kwalificeren. Vanuit het marketingstandpunt is het begrijpelijk, dat zo'n veelzijdig rekenprogramma, waaraan zeker 2 jaar research en arbeid is voorafgegaan, tegen een zo "gunstig" mogelijke prijs op de markt wordt gebracht. Toch willen de makers ervan door kopieerbeveiligingen aan te brengen illegale distributie tegengaan. Er klopt dus iets niet aan de marketing! Wanneer het programma zo veelzijdig is, dat iedereen het zal willen gebruiken, ligt het voor de hand, dat het pakket zichzelf niet hoeft te bewijzen en dus tegen een veel lagere prijs (waardoor illegale distributie, ook wel "kopieren" genoemd, minder zinvol wordt) op de markt had kunnen worden gebracht. Een goed voorbeeld van eigen bodem is een administratief pakket, dat door een centraal gelegen softwarehuis wordt

aangeboden. Dit pakket is niet beter of slechter dan vergelijkbare en in overvloed aanwezige financiële bedrijfspakketten, doch wordt voor de kleine APPLE drives uitgebracht in "beveiligde versie". Wanneer het pakket zo goed zou zijn, dat elk bedrijf ermee zou kunnen werken, zou het tegen zo'n gunstige prijs kunnen worden aangeboden, dat illegale distributie niet noodzakelijk zou zijn. Dat beide voorbeelden in het illegale distributiecircuit zijn toont

- a) dat de bewuste programmatuur goed in de markt ligt, doch te duur wordt bevonden en
- b) dat velen hun software "op maat" wensen te kunnen maken.

Dat laatste is veel betekenend: wat heeft men in een bedrijf aan "micro" toepassingen op een "macro" schaal - waarom is deze software afgestemd op gebruik in de hobbysfeer, dus voor kleine diskette-systemen ? Als wij deze redeneertrant voortzetten, komen we niet tot de essentie van het probleem. De oorzaak van het bestaan van een kopieerprobleem ligt in de afschuw, die systeem-ontwerpers hebben van software-implementatie in hun systemen. Computerbouwers zijn electronici - ze bouwen ingewikkelde stroomcircuits, die een logische functie moeten hebben. Wat deze componenten (meestal IC's) voor mogelijkheden bieden voor het maatschappelijk leven deert hen niet of nauwelijks! Hierin schijnt vooralsnog ook geen wezenlijke kentering te komen. Deze sterke conclusie krijgt een diepere betekenis, wanneer we beseffen, dat computersoftware uit maar 4 stromingen bestaat:

- 1-de recreatieve programmatuur, w.o. spelletjes/educatie
- 2-de tekstverwerking, uitbouw van de typemachine
- 3-de gegevensverwerking, uitbouw van de kaartenbak
- 4-de getallenverwerking, uitbouw van het rekenwerk

Dit zijn naar onze mening de 4 wielen van de computermobiel...

Wie kan echter aangeven wie de OORSPRONKELIJKE ontwerper van verwerkingsroutines was ten behoeve van de vier opgesomde doelen?

Wie kan bijgevolg "authenticiteit" claimen op zijn product, dat toegesneden is op de vier opgesomde deelmarkten ? Een voorbeeld maakt dit vraagstuk duidelijk: kan een romanschrijver een andere schrijver betichten van inbreuk op zijn auteursrecht doordat de andere schrijver een roman het licht doet zien, waarin niet Antje achter Jantje, maar Mientje achter Pietje aanzit ? Ook al speelt dit tafereel zich respectievelijk aan de Noord- en de Zuidpool af?

Het boekenvak houdt op te bestaan, wanneer deze vraag bevestigend zou worden beantwoord.

Een andere zaak is het, wanneer het gehele werk klandestien wordt gekopieerd en verhandeld.

Waarom kopiëren boekhandelaren hun boeken niet en waarom zouden computerleveranciers hun software wel kopiëren ? Dat de electronica op voet van oorlog leeft met de gebruikerswensen is hiermee voldoende aangetoond. Het houdt geen gelijke tred en tussen de wrijvende krachten leeft de software-pionier.

Is hij een "born loser" ? Pas wanneer software IN het computersysteem aanwezig is en modulair te vormen is tot individueel gewenste werkvormen valt deze vraag positief te beantwoorden. De tijd zal dit ons leren. Computergebruikers zullen overigens moeten oppassen, dat de software-wereld niet remmend gaat werken op mogelijkheden, die nu al aan de horizon zijn verschenen. Tot het zover is, dat software tot firmware wordt geïntegreerd, zullen we gewoon moeten leren leven met een ontwikkelingsstuip. Termen als "piraterij" en "diefstal" doen in dat licht grotesk aan, want hij die van schulden vrij is werpe de eerste steen!

Een veelbelovend alternatief is het UITGEVEN van software. Hierbij wordt programmatuur aan de gebruiker geleverd tegen kostprijs, doch met een relatief duur en zeer uitgebreid handboek erbij. Indien de programmatuur daarvoor van voldoende kwaliteit is, zal een gebruiker misschien de programmatuur malefide willen kopiëren, doch op grond van de prijs geen been zien in het kopiëren van het handboek. Het grootste voordeel van deze aanpak is de stelselmatige schifting, die er ten gunste van de gebruiker uit voortkomt. Kwalitatief slechte software zal bij een noodzakelijke, uitgebreide beschrijving ervan al a priori door de mand vallen. Resultaat: gebrekkige software wordt zeer goedkoop, kwalitatief hoogstaande programmatuur wordt goedkoper. Het uitgeven van goede software kan de komende tien jaar een aantrekkelijke zaak worden.

De computerwereld kan lering trekken uit ervaringen in de uitgeverij, of kan daarvoor een verbond met gespecialiseerde uitgevers aangaan. Het kan zeker geen kwaad, wanneer het mystieke waas rond het vak van programmeur wordt opgeheven en hij de status krijgt van "software-auteur". Vooral juridisch zal zijn positie dan globaal versterkt worden!

4. TEN BESLUIITE

De ontwikkelingen op het gebied van microprocessoren gaan snel.

Het vervult de gebruiker van de APPLE II weleens met onrust, of zijn dierbare "machine" over enkele jaren nog zal meetellen. Degeen, die de aanschaf van een microcomputer overweegt, heeft het met deze vraag so-wie-so al moeilijk. Ook wij kunnen niet in de toekomst kijken. Zeker is het, dat APPLE COMPUTERS INC. in weinig jaren met de succesvolle APPLE II microcomputer voor een standaard heeft gezorgd, waarmee velen zich (tevergeefs) trachten te meten.

De kracht van het APPLE computersysteem kun je pas waarden, wanneer je ermee hebt gewerkt. Merkwaardig is het evenwel, dat de fabrikant nooit moeite heeft gedaan om kleine tekortkomingen van de populaire APPLE II op te heffen. Vrijwel alle "verbeteringen" zijn buiten APPLE COMPUTERS INC. tot stand gekomen. Het aantal producenten van accessoires voor de APPLE II is overweldigend en getuigt van het grootse sukses. Ongetwijfeld kan onze APPLE II nog veel jaren mee. Het verschijnen van de minder geslaagde APPLE III en de aankondiging van de APPLE IV doen daar niets aan af.

De laatstgenoemde computers zijn dure business systemen, die niet bestemd zijn voor "gedecentraliseerd" gebruik. De APPLE II is in principe een desktop computer, die overal voor is in te zetten en overal mee naar toe kan worden genomen. Een fenomeen vormt de juist bij het ter perse gaan van dit handboek verschenen "PEAR II". Deze microcomputer is eigenlijk een tweemaal zo grote versie van de APPLE II. Voor de zuiver professionele gebruiker van het APPLE systeem biedt de "PEAR" meer mogelijkheden voor een zwaarder gebruik. Zo heeft de "PEAR" een zeer goed alfanumeriek toetsenbord met los numeriek keypad. Voorts heeft hij een zwaardere voeding en biedt hij ruimte voor 14 peripherals. In hoeverre APPLE een "peer" naast zich zal gedogen, valt te bezien, ofschoon kwade tongen beweren, dat de PEAR II een APPLE-produkt is! Een tamelijk identiek product vormt de "Black APPLE"; een door Bell & Howell in licentie vervaardigde APPLE II, ook met numeriek toetsenbordje. Opvallend is, dat deze produkten op de Benelux markt verschijnen, nu de computerhandel ontevreden geluiden laat horen over de marketing van APPLE computers.

EEN EVALUATIE ANNO VOORJAAR 1982

Wij ontkomen er niet aan, om in het licht van de verschijning van twee sterke alternatieven voor de APPLE II microcomputer aan U onze overdenkingen voor te houden.

Vast staat voor ons, dat APPLE COMPUTER INC. voor een rotsvaste "standaard" heeft gezorgd met haar APPLE II computers en in veel mindere mate door de mislukte APPLE III. Dat laatste buiten de schuld van APPLE.

Dat APPLE faalt, waar het de marketing van haar systemen naar Europa betreft, spreekt uit de verkoopcijfers, die het instituut van marktonderzoek uit Boston (VS) recentelijk publiceerde.

Wanneer wij de verkoop cijfers van "De Grote Drie" bekijken dan valt op, dat Comodore (PET) met de verkoop van 106.000 eenheden op de Europese markt de leiding heeft met een markt aandeel van 30-50% afhankelijk van de aangelegde criteria.

Apple volgt met 32.600 exemplaren en Tandy sluit de rij met 26.000 verkopen. Zo op het oog valt APPLE niet uit de toon, maar binnen het marktgebied van haar computer deelt zij de laagste markt met 64 (!) directe concurrenten... In Amerika noemt men een fabrikant die zo ingeklemd zit tussen concurrenten helaas "Mr. Nobody".

Een feit is het, dat handelaren in de Benelux zeer ontevreden zijn over de marketing van Apple's, maar vol lof zijn over de eigen Apple-Service Organisatie die in Zeist is gehuisvest. Maar het gaat niet primair over de service, wel over de verkopen...

Inmiddels zouden er in Cupertino bij Apple pittige brieven liggen over de schromelijk te kort schietende marketing.

Een volgend feit is dat APPLE niets heeft gedaan om de algemeen gevoelde gebreken van de Apple II op te vangen. Geen upper en lowercase, geen Euro-Color en RGB, geen betere voeding, geen dikker moederbord en geen 80 karakters op een regel. Dit heeft men overgelaten aan derden, waardoor Apple een erg rommelige indruk vestigt. Hiernu hebben de Europese fabrikanten op ingespeeld. De "Nederlands/Engels/Belgische" PEARCOM en de "Duitse" BASIS 108 serie tonen aan, dat er een markt is voor een sterk verbeterde versie van de Apple II. In prijs liggen deze alternatieven gelijk met de standaard 48K APPLE II. Dus de keus wordt voor veel eisende, meer professioneel ingestelde gebruikers erg gemakkelijk gemaakt.

DE "PEARCOM" MICRO-FRAME

=====

Werken met de nieuwe "PEARCOM" micro-frame is een eigenaardige ervaring. Kenmerkend is, dat deze microcomputer weliswaar werd uitgerust met een 6502 processor, doch dat daarvoor geen besturingsprogramma, zelfs geen "monitor" bij aanschaf mee komt. Je kunt daarom kiezen uit een aantal aantrekkelijke opties. Tegen een geringe meerprijs kun je de bekende APPLESOFT ROM's laten installeren, waardoor de "Peer" bij het inschakelen als een echte "Apple" opkomt:

biep APPLE II

Of je kunt er gelijk een Z-80A kaart, de softcard van Microsoft kopen en gelijk professioneel werken met CP/M en MBASIC (GBASIC). De laatste mogelijk ligt het meest voor de hand, omdat Applesoft niet werd geschreven om daarmee snelle en ingewikkelde diskbesturing toe te passen. in feit ontbreken de commando's daarvoor (in tegenselling tot SAVE, S(H)LOAD via de cassetteband!) en moet je je met de Apple II behelpen met een ingewikkeld Disk Operating System. De professionel uitvoering van de "PEARCOM" met zijn prachtige toetsenbord, mooi ergonomisch geplaatste toetsen en het numerieke keypad vraagt om een bedrijfsmatige, althans professionel benadering.

Het miniale prijsverschil dat tussen de "PEARCOM" en de "APPLE II" bestaat (nauwelijks 500 gulden 35 Bfr) zal het de meer professioneel kwaliteiten zoekende koper erg gemakkelijk maken.....

De geheugenorganisatie van de "PEARCOM" is zondermeer uitgekiend. Er is 96K RAM werkgeheugen beschikbaar, waarvan boven de 32K maar liefst 4 te selecteren banken. Omdat een 8-bit microprocessor slechts 65535 adressen kan overzien, zal hij nooit meer dan 64K ineens kunnen bewerken. Vandaar de "bank select", die met een zelfde POKE is software is te besturen. Door snel te POKEn kun je de illusie hebben toch met 96K van doen te hebben! Dit noemt men een "transparante" organisatie.

Het Hex adres \$C060 (bij de APPLE II de cassetterecorder) beheerst het totale schakelbeeld binnen de "PEARCOM". Het woord "lumineus" is hierbij op zijn plaats.

functie	locatie	b7	b6	b5	b4	b3	b2	b1/b0
write	-16288	x	x	kar.	low.	ex	io	bank Cn
aan				set	case	vid	sel	select
read	-16288	cass	x	norm	up	norm	io	norm.
uit		in		set	case	video	i	bank

Schakellocatie \$C060/-16288 macht 10

Willen wij bij de "PEARCOM" desnoods nog een 256K RAM-kaart aan brengen dan is dat mogelijk.

Er zijn voor I/O toepassingen 2x7 slots aanwezig, in 2 banken, die ook via POKE -16288 worden geswitched.

De "PEARCOM" kan standaard worden voorzien van 6 type 2716 INTEL EPROM's met besturingssoftware (FORTRAN, COBOL of BASIC), of (jumper omzetten) met 2316 ROM's van het bekende Aplesoft type.

Je kunt een en ander niet door elkaar gebruiken.....

Door de merkwaardige memorie-mapping met de drie ingebouwde "RAM-kaarten" is het mogelijk gebruik te maken van maar liefst 10 verschillende tekst- en lowresolution graphic screens. In hi-res slechts zes!!!

Het is dus theroretisch mogelijk door "page-flipping" een complete tekenfilm op de "PEARCOM" te runnen.

In de tekst-mode is het bekende "APPLE II" stramien beschikbaar: 960 karakters in normal, inverse en flashing mode. Standaard is het beeld Uppercase (net als bij de APPLE hoofdletters), of Lowercase. Er zijn echter ook Griekse en grafische tekens beschikbaar in extended video mode. Wie eigen karakters (bv gemixte upper- en lowercase) wil toepassen moet zelf zijn EPROM maken en dat kan heel gemakkelijk.

Ook standaard bezit de "PEARCOM" een PAL-kleuruitgang voor aansluiting op een kleurentelevisie. Het gaat prima maar het is natuurlijk niet erg fraai. Een RGB interface is inmiddels beschikbaar.

Samenvatting

=====

Indien de "PEARCOM" met Applesoft ROM's of AS-kaart of een 16K RAM-kaart wordt uitgerust, is hij "identiek" aan de Apple II computer. Dit handboek is dan ook aanbevolen voor de "PEARCOM".

Wel dient men rekening te houden met de vele extra's van de "PEARCOM", die veel "tekortkomingen" van onze APPLE II teniet doen. Maar goed verschil moet er zijn!

Het is zeker niet zo, dat APPLE-gebruikers zich beconcurrered moeten voelen. Er komt alleen een stevig uitgebouwd vriendje bij. Alle standaard-APPLE software werkt op de "PEARCOM"

Werken met de Z-80A Softcard

Inleiding

De door het Amerikaanse systeemhuis "Microsoft" op de markt gebrachte Plug-in kaart met de Z-80A processor is waarschijnlijk een van de meest revolutionaire ontwikkelingen rond de Apple-computer.

In een schitterend pakket wordt deze "Softcard" op de markt gebracht. Erbij mee komen de fysieke kaart, een tweetal 5.25 inch diskettes met CP/M en twee versies van het ANSI genormeerde BASIC (M en G), versie 5.0. Voorts worden twee keurig verzorgde, dikke handleidingen mee geleverd, die alle in's en out's van het Z-80A systeem uit de doeken doen.

Een en ander betekent, dat alle professionele software, die in zo ruime mate voor de Z-80 beschikbaar is, nu ook op de Apple verwerkt kan worden.

Wat is CP/M

CP/M is in tegenstelling tot wat velen denken- GEEN Computertaal. CP/M betekend "Control Program/Monitor" en werd in 1973 door Gary Kildall INTEL ontwikkeld. Het is een disk-operating-systeem.

CP/M is het sukses verhaal van een fabrikant van computer peripherals (m.n. diskdrives), die gekoppeld moesten worden aan allerlei computer frames. Toen dat goed gelukt was, kwam de software ontwikkeling opgang, die CP/M maakte tot wat het nu is: een hoog ontwikkeld disk-operating-systeem met hulp programma's. De CP/M versies 1.4 en 2.2 zijn thans het wijdst verbreid.

Het installeren van de softcard.

Meer dan anders is het voor CP/M nodig, dat bepaalde peripherals (printers, 80-kolomkaart, diskdrives) op genormeerde slots worden aangebracht. Zo hoort de printer op slot 1, diskdrives op slot 6. De Softcard kan in slot 7 of slot 4. Bij het monteren van de kaart schakelen we de Apple uit, lichten het deksel op en steken de kaart op zijn plaats. Deksel dicht en klaar!

Het kopiëren van de masterdiskette.

Het is verstandig de meegeleverde CP/M masterdiskette te kopiëren. Daartoe doen wij de masterdisk (in 13 of 16 sector formaat, beide versies zijn aanwezig) in de diskdrive. Vervolgens tikken we pr#6 om op te booten. Binnen weinige tellen staat het CP/M logo op het scherm. Nu kunnen we ons de eerste beginselen van dit systeem leren eigen maken. Het is even wennen aan de afwezigheid van een cursor. Op het beeldscherm zie je inplaats van het bekende Applesoft haakje het volgende:

A >

Dit betekend dat CP/M klaar is voor de ontvangst van je eerste commando.

In tegenstelling tot het Apple-DOS kent CP/M geen drive 1, 2 of 3. Het kent slechts drive A:, B:, en C:, vandaar de stickers die op je Z-80 pakket zitten.

Tik nu eens DIR en wacht af wat er na een return te zien is. DIR is het CP/M commando door Directory of in Apple-DOS jargon "Catalog".

Ondermeer zie je het programma "FORMAT" (met een achtervoegsel COM) staan. Om een kopie te maken van de master zullen wij van dit programma het eerste gebruik moeten maken.

Tik daarom FORMAT B: (of FORMAT A: als je 1 drive hebt). Doe een ongebruikte diskette in drive B: en tik op de return toest, zodra het programma zich aandient. Binnen een tiental tellen is de nieuwe diskette "geformatteerd". Na het formateren tik je nogmaals return om weer in de uitgangspositie terug te komen. Degenen met 1 drive moeten tussentijds diskette wisselen. Let goed op en kijk wat het programma verwacht!

Nu gaan we master werkelijk kopiëren.

A > COPY B:=A:

Ofwel CP/M kopieer voor mij op drive B: de programma's van drive A:! Zonder veel moeite wordt dit proces afgerond, maar..... LET OP. Je hebt zo geen kopie van de master, omdat het feitelijk Boot-CP/M nog ontbreekt.

Daartoe tikken we opnieuw:

A > COPY B:-A:/S /S betekent System, ofwel kopieer nu de CP/M boot-routine.

PAS NU ZAL ER EEN GEREDE KOPIE BESTAAN VAN DE MASTER!

Het is mogelijk deze aan te passen aan een Apple met een Language- of RAMkaart. Berg nu de master-diskette veilig op en doe de nieuwe werkkopie in drive A: tik (als je een ramkaart hebt) vervolgens:

CPM56 A:

en de computer doet de rest voor je.

BASIS-108: HET NIEUWSTE ALTERNATIEF

Voortbordurend op het sukses van de APPLE II micro komen nu in snel tempo microcomputers op de Europese markt, die volmaakt "Apple compatible" zijn. Opvallend is, dat deze alternatieven voor de APPLE verschijnen op een moment, dat de handel uitgekeken raakt op de wijze, waarop Apple met haar product omgaat. Wij maakten daarvan al eerder gewag.

Het nieuwste alternatief voor de APPLE II is de daaraan volmaakt compatibele "BASIS-108" microcomputer uit Duitsland. Evenals de "PEARCOM" lost dit alternatief een groot aantal typische (en onnodige) tekortkomingen van de standaard APPLE II op

In feite is de "BASIS-108" een type uit een reeks van drie. Al naar gelang de uitbouw (tot en met twee Apple mini diskdriuves toe) kent men de BASIS 108, 208 en 216.

Karakteristiek is het uiterlijk van de BASIS-108. Hij lijkt precies op de APPLE III ! Een zeer uitgebreid en fraai toetsenpaneel (je durft het geen bord meer te noemen) en een doosvormige kast, waarin de processoren (meervoud) zitten met geen, een of twee diskdrives. De BASIS-108 bezit een Z-80A en een 6502 processor hetgeen ongelofelijke mogelijkheden inhoudt.

Standaard wordt hij uitgerust met 64K RAM werkgeheugen, die uitgebouwd kan worden tot 128K

Hij heeft 10K ROM vrij en kan willekeurig worden uitgerust met de bekende Applesoft ROM's en-of eigen EPROM's met bv. FORTRAN, COBOL, etc. In dit opzicht heeft deze machine overeenkomsten met de "PEARCOM".

Maar, waar de "PEARCOM" tobt met de PAL-kleuren, biedt de BASIS 108 juist iets meer: een standaard RGB-uitgang, die bij de PEARCOM optioneel is.

Het beeldscherm is omschakelbaar van 80 karakters per regel naar 40 uitgaande van 24 regels per scherm. Dus professionele tekstverwerking is mogelijk.

Hij bezit een cassette recorder aansluiting "maar" zes slots, waarin de peripherals gaan. Gezien het feit, dat deze BASIS-microcomputer zo volledig is, zal er echter maar weinig "in behoeven worden gestoken". Want ook standaard zijn: een parallel-interface en een V-24 seriële interface.

Voor het overige is het een en al APPLE

Werken met MBASIC-80 versie 5.0

Met CP/M en de Z-80 kaart komen twee versies voor van Microsoft's MBASIC-80.

De versie M kent geen HI-RES commando's en de versie G wel.

Je kunt MBASIC booten door vanuit CP/M te tikken:

A > MBASIC

Het MBASIC is volgens ANS I normblad BSRX3. 60-1978 gestandariseerd. Programma's in M(G)Basic werken daarmee op andere MBASIC-systemen.

Op nogal wat punten is MBASIC afwijkend van APPLESOFT. Zo kent het de begrippen als COMMON, CHAIN, PRINT, USING, IF....THEN..WHILE....WEND, AUTO en RENUM.

Krachtig zijn de diskcommando's die niet meer in het omslachtige "PRINT CHR\$(4)" behoeven te worden verpakt.

Toch kunnen Applesoft programma's (zonder 6502 PEEKs en POKEs) worden geconverteerd naar MBASIC. Erg aantrekkelijk is voorts het werken met dubbele precisie. Je hebt nu de keus uit gehele getallen, enkelvoudige precisie (7 cijfers achter de komma) en dubbele precisie (16 cijfers achter de komma).

Probeer:

```
10 A%25.34
20 PRINT A%
```

```
RUN
25
```

```
10 A#6=# /7
20 PRINT A
30 PRINT A#
```

```
RUN
.857143
.8571438571428571
```

Aantrekkelijk nietwaar?

Alle wiskundige functies worden ondersteund, zoals bij Applesoft het geval is. Het kan daarom zeker geen kwaad eerst goed in Applesoft te leren Basic programmeren. Door te tikken SAVE "PROG" kun je een programma naar diskette wegschrijven. Wil je dat vanuit drive A: doen naar drive B:, tik dan SAVE "B:PROG".

Benamingen van programma's of tekstfiles mogen in CP/M niet langer zijn dan 8 tekens, exclusief een drie-letterig achtervoegsel, bv. COM of BAS. Het is zeker de moeite waard met de Z-80 kaart te experimenteren en te vergelijken met het 6502-systeem.

```

10  REM  TRICKDOS PROGRAMMA
20  TEXT : HOME : POKE - 16298,0: POKE - 16300,
    0: POKE - 16368,0
30  GOSUB 2010: GOSUB 3010: GOSUB 2510: GOTO 610
100  REM                                PEEK
    COMMANDTABLE                        AND CREATE ARRAY
102  REM  ARRAY DOS$(R1-28,C1-2)      C1=COMMAND
    C2=START ADDR
104  REM  DOS#=DOS COMMAND TABLE
106  REM  DOS=ADDR COMMAND TABLE
110  TM = START
120  IF PEEK (TM) = 0 THEN FIN = TM - 1: GOTO 14
    0: REM  FIND END OF TABLE
130  TM = TM + 1: GOTO 120
140  I = 1: FOR J = 1 TO 29: FOR K = 1 TO 2:DOS$(J
    ,K) = "": NEXT K,J:DOS$(1,2) = STR$ (START)
    :DOS$ = "": REM  INITIALIZE
150  FOR DOS = START TO FIN
160  IF PEEK (DOS) > 127 THEN DOS$(I,1) = DOS$(I
    ,1) + CHR$ ( PEEK (DOS)):DOS$ = DOS$ + CHR$
    ( PEEK (DOS)):DOS$((I + 1),2) = STR$ (DOS +
    1):I = I + 1: GOTO 180: REM  IF HI BYTE INC
    REMENT I
170  DOS$(I,1) = DOS$(I,1) + CHR$ ( PEEK (DOS)):D
    OS$ = DOS$ + CHR$ ( PEEK (DOS))
180  NEXT DOS: RETURN
300  REM  DEC-->HEX
310  HD% = DOS / 256:NBR = HD%: GOSUB 340:HB$ = HE
    X$
320  LD% = FN MOD(DOS):NBR = LD%: GOSUB 340:LB$ =
    HEX$
330  HEX$ = HB$ + LB$: RETURN
340  H% = NBR / 16 + 1:L% = NBR / 16:L = L% * 16:L
    % = NBR - L + 1
350  HEX$ = MID$ (H$,H%,1) + MID$ (H$,L%,1): RETURN

400  REM  REORGANIZE COMMAND TABLE
410  IF PT = 1 THEN T1$ = "": GOTO 430
420  T1$ = LEFT$ (DOS$, VAL (DOS$(PT,2)) - START)

430  FOR I = 1 TO LEN (NC$):T2$ = T2$ + MID$ (N
    C$,I,1): NEXT
440  IF PT = 28 THEN T3$ = "": GOTO 460
450  T3$ = RIGHT$ (DOS$,FIN + 1 - VAL (DOS$((PT +
    1),2)))
460  DOS$ = T1$ + T2$ + T3$:T2$ = ""
470  DOS = START
480  FOR I = 1 TO LEN (DOS$): POKE DOS, ASC ( MID$
    (DOS$,I,1)):DOS = DOS + 1: NEXT
490  FIN = FIN + LEN (NC$) - LEN (OC$)
500  POKE FIN + 1,0: RETURN
600  REM  MENU
610  HOME:TT$ = "=====": GOSUB 3110
620  TT$ = "TRICK DOS MENU": GOSUB 3110
630  TT$ = "=====": GOSUB 3110
640  VTAB 6: PRINT "1.DISPLAY CURRENT DOS COMMAND
    TABLE.": PRINT

```



```

650 PRINT "2.CHANGE DOS COMMAND TABLE.": PRINT
660 PRINT "3.RESTORE NORMAL DOS COMMAND TABLE.":
  PRINT
670 PRINT "4.TRY SANDY'S COMMAND'S.": PRINT
680 PRINT "5.EXIT PROGRAM.": PRINT : PRINT
690 VTAB 17: CALL - 958: PRINT "    WHICH CHOICE?" : GET I$: PRINT I$:CH = VAL (I$)
700 IF CH < 1 OR CH > 5 OR I$ = "" THEN 690
710 ON CH GOTO 800,1000,1300,1300,1500
800 REM DISPLAY CURRENT TABLE
810 HOME :TT$ = "=====": GOSUB 3110
820 TT$ = "CURRENT DOS COMMANDS & ADDRESSES": GOSUB
  3110
830 TT$ = "=====": GOSUB 3110
840 IF NOT FF THEN VTAB 8: INVERSE :TT$ = "READING
  DOS COMMAND TABLE": GOSUB 3110: NORMAL

850 GOSUB 110: VTAB 4: CALL - 958
860 PRINT : HTAB 2: INVERSE : PRINT "DEC": HTAB
  8: PRINT "HEX": HTAB 22: PRINT "DEC": HTAB
  28: PRINT "HEX": NORMAL : PRINT
870 FOR I = 1 TO 14
880 PRINT DOS$(I,2) " ":DOS = VAL (DOS$(I,2)): GOSUB
  310: PRINT HEX$ "DOS$(I,1)": HTAB 21: PRINT
  DOS$((I + 14),2) " ":DOS = VAL (DOS$((I + 1
  4),2)): GOSUB 310: PRINT HEX$ "DOS$((I + 14
  ),1): NEXT
890 IF FF THEN FOR I = 1 TO 5: PRINT : NEXT : RETURN

900 VTAB 22: PRINT "LIST TABLE TO PRINTER (Y/N)
  ? " : GET I$
909 REM *** DOS$(20,1):X          IN 910 IS PRINT
  ERSLOT
910 IF I$ = "Y" THEN FF = 1: HTAB 1: CALL - 998
  : CALL - 958: PRINT B$: INVERSE : PRINT "TURN
  ON PRINTER AND PRESS ANY KEY": PRINT : HTAB
  10: PRINT "EXPECT A PAUSE " : GET I$: PRINT
  : NORMAL : PRINT D$:DOS$(20,1):1: GOSUB 810:
  FF = 0:PRI
920 IF I$ = "N" THEN 610
930 HTAB 1: GOTO 900
1000 REM CHANGE TABLE
1010 HOME :TT$ = "=====": GOSUB 3110
1020 TT$ = "CHANGE COMMANDS": GOSUB 3110
1030 TT$ = "=====": GOSUB 3110
1040 VTAB 4: CALL - 958: VTAB 8: INVERSE :TT$ =
  "READING DOS COMMAND TABLE ": GOSUB 3110: NORMAL

1050 GOSUB 110: VTAB 5: CALL - 958
1060 FOR I = 1 TO 7
1070 PRINT DOS$(I,1): HTAB 10: PRINT DOS$((I +
  7),1): HTAB 20: PRINT DOS$((I + 14),1): HTAB
  30: PRINT DOS$((I + 21),1): NEXT
1080 VTAB 14: CALL - 958: INPUT "TYPE COMMAND TO
  BE CHANGED: ":OC$: IF OC$ = "" THEN 1180
1090 OC$ = MID$(OC$,1, LEN (OC$) - 1) + CHR$ (
  ASC ( RIGHT$(OC$,1)) + 128): REM TURN HI
  BIT ON IN LAST LETTER OF COMMAND
1100 FOR I = 1 TO 28: IF OC$ = DOS$(I,1) THEN PT
  = I: GOTO 1130 REM PT=POINTER TO POSITION
  OF COMMAND IN ARRAY

```

```

1110 IF I = 28 THEN PRINT B$: VTAB 16: INVERSE
    : PRINT " NOT A VALID CURRENT COMMAND ": NORMAL
    : FOR J = 1 TO 3000: NEXT : GOTO 1080
1120 NEXT I
1130 VTAB 16: CALL - 958: INPUT "TYPE NEW COMMA
ND: ";NC$: IF NC$ = "" THEN 1130
1140 NC$ = MID$(NC$,1, LEN (NC$) - 1) + CHR$ (
ASC ( RIGHT$(NC$,1)) + 128): REM TURN ON
HI BIT IN LAST LETTER OF COMMAND
1150 PRINT B$: VTAB 18: HTAB 3: PRINT "CONFIRM (
Y/N) ? " : GET I$: PRINT I$
1160 IF I$ = "Y" THEN VTAB 20: INVERSE : PRINT
" WRITING COMMAND TABLE ": GOSUB 410: VTAB 1
8: HTAB 1: CALL - 958: PRINT " CHANGE COMPL
ETED ": NORMAL : GOTO 1220
1170 IF I$ < > "N" THEN VTAB 18: CALL - 958: GOTO
1150
1180 VTAB 18: CALL - 958: PRINT : PRINT "RETURN
TO MENU OR TRY AGAIN (M/A)? " : GET I$: PRINT
I$
1190 IF I$ = "A" THEN GOTO 1080
1200 IF I$ = "M" THEN 610
1210 GOTO 1180
1220 VTAB 20: CALL - 958: PRINT "ANOTHER CHANGE
(Y/N) ? " : GET I$: PRINT I$: IF I$ = "Y" THEN
1040
1230 IF I$ = "N" THEN 610
1240 GOTO 1220
1300 REM RESTORE NORMAL TABLE OR INSTALL SANDY'
S TABLE
1310 VTAB 20: INVERSE : PRINT " WRITING COMMAND
TABLE ":
1320 NDOS$ = "":MYDOS$ = ""
1330 FOR I = 1 TO 132: READ D:NDOS$ = NDOS$ + CHR$
(D): NEXT
1340 FOR I = 1 TO 67: READ D:MYDOS$ = MYDOS$ + CHR$
(D): NEXT : RESTORE
1350 DOS = START
1360 IF CH = 3 THEN TM$ = NDOS$:TT$ = "NORMAL DO
S COMMAND TABLE REESTABLISHED ":FIN = START +
LEN (NDOS$) - 1
1370 IF CH = 4 THEN TM$ = MYDOS$:TT$ = " SANDY'S
COMMAND TABLE INSTALLED ":FIN = START + LEN
(MYDOS$) - 1
1380 FOR I = 1 TO LEN (TM$): POKE DOS, ASC ( MID$
(TM$,I,1)):DOS = DOS + 1: NEXT
1390 POKE FIN + 1,0
1400 HTAB 1: PRINT TT$: NORMAL : GOSUB 3210: HTAB
1: GOTO 690
1500 REM END PROGRAM
1510 POKE - 16298,0: POKE - 16300,0: POKE - 1
6368,0: TEXT : HOME
1520 VTAB 10: INVERSE :TT$ = " END OF TRICK DOS
PROGRAM ": GOSUB 3110: NORMAL
1530 VTAB 15: PRINT " INITIALIZING A DISK BEFORE
REBOOTING": PRINT " WILL PRESERVE THE CURRE
NT DOS COMMANDS"
1540 VTAB 22: END
2000 REM INITIALIZE
2010 DIM DOS$(30,2)

```



```

2020 D$ = CHR$ (4):B$ = CHR$ (7):SS$ = "
      ": REM 21 SPACES

2030 H$ = "0123456789ABCDEF"
2040 DEF FN MOD(X) = X - INT (X / 256) * 256: REM
      SIMULATE MOD FUNCTION
2050 START = 43140: REM START OF TABLE
2060 RETURN
2100 DATA 73,78,73,212,76,79,65,196,83,65,86,197
      ,82,85,206,67,72,65,73,206,68,69,76,69,84,19
      7,76,79,67,203,85,78,76,79,67,203,67,76,79,8
      3,197,82,69,65,196,69,88,69,195,87,82,73,84,
      197,80,79,83,73,84,73,79,206,79,80,69,206,65
      ,80,80,69,78,196
2110 DATA 82,69,78,65,77,197,67,65,84,65,76,79
      ,199,77,79,206,78,79,77,79,206,80,82,163,73,
      78,163,77,65,88,70,73,76,69,211,70,208,73,78
      ,212,66,83,65,86,197,66,76,79,65,196,66,82,8
      5,206,86,69,82,73,70,217: REM NORMAL TABLE

2120 DATA 73,170,76,196,83,214,82,85,206,67,72,
      206,68,204,76,203,85,76,203,67,211,82,196,69
      ,88,195,87,210,80,83,206,79,208,65,203,82,69
      ,206,67,65,212,77,206,78,77,206,80,163,73,16
      3,77,65,216,70,208,73,78,212,66,211,66,204,6
      6,210,86,69,210
2130 DATA 77,206,78,77,206,80,163,73,163,77,65,
      216,70,208,73,78,212,66,211,66,204,66,210,86
      ,69,210: REM SANDY'S TABLE
2500 REM INSTRUCTIONS
2510 HOME :TT$ = "===== ": GOSUB 3110
2520 TT$ = "INSTRUCTIONS": GOSUB 3110
2530 TT$ = "===== ": GOSUB 3110
2540 VTAB 7: CALL - 958: PRINT "DO YOU WANT INS
      TRUCTIONS (Y/N) ? ": GET I$: PRINT I$: IF I
      $ = "N" THEN RETURN
2550 IF I$ < > "Y" THEN 2540
2560 POKE 34,4: VTAB 5: CALL - 958
2570 PRINT "1.THE DOS COMMAND TABLE RESIDES AT R
      AM": PRINT "XLOCATIONS $A884 TO $A908 (DEC 4
      3140": PRINT "XTO 43272).": PRINT
2580 PRINT "2.EACH COMMAND IS REPRESENTED BY ASC
      II": PRINT "XCHARACTER CODES. ONLY THE LAST
      LETTER": PRINT "XOF A COMMAND HAS THE HIGH B
      IT ON SO": PRINT "XTHAT DOS CAN RECOGNIZE TH
      E END OF THE"
2590 PRINT "XCOMMAND. NOTE THE EXAMPLES BELOW:":
      PRINT : PRINT "LOAD = 4C 4F 41 C4": PRINT
      "INIT = 49 4E 49 D4": PRINT "RUN
      = 52 55 CE": PRINT : PRINT
2600 PRINT "3.ZERO MARKS THE END OF THE TABLE."
2610 GOSUB 3210: HOME
2620 PRINT "4.THIS PROGRAM WILL ENABLE YOU TO AL
      TER": PRINT "XTHE COMMAND TABLE. YOU MAY DES
      IRE TO": PRINT "XCHANGE 'CATALOG' TO ": PRINT INVERSE
      : PRINT "CAT": PRINT "XNORMAL : PRINT "XOR 'SAVE' T
      O ": PRINT "X": PRINT INVERSE : PRINT "SV": PRINT NORMAL

```



```

2630 PRINT ".BE SURE THAT NEW DOS COMMAND": PRINT
" DOES NOT DUPLICATE THE FIRST PART OF": PRINT
"AN APPLESOFT BASIC COMMAND, OTHERWISE": PRINT
"AN UNUSUAL EVENTS MAY OCCUR. EXPERIMENT!"
2640 PRINT "4.TIREDNESS OR SILLINESS MAY RESULT I
N": PRINT "XWEIRD SYMBOLS!!!": PRINT
2650 PRINT "5.THESE MODIFICATIONS WILL TRIGGER A
": PRINT "XSYNTAX ERROR IF A DIRECT OR DEFER
RED": PRINT "XCOMMAND UTILIZES 'NORMAL' TERM
INOLOGY."
2660 PRINT "6.": INVERSE : PRINT "TRICK DOS": NORMAL :
2670 POKE 34,0: GOSUB 3210: RETURN
3000 REM TITLE PAGE
3005 REM SF APPLE CORE FORMAT
3010 INVERSE : VTAB 4
3020 TT$ = SS$: GOSUB 3110: GOSUB 3110
3030 TT$ = " TRICK DOS ": GOSUB 3110
3040 TT$ = SS$: GOSUB 3110: GOSUB 3110
3050 TT$ = " BY SANDY MOSSBERG ": GOSUB 3110
3060 TT$ = SS$: GOSUB 3110: GOSUB 3110: NORMAL
3070 VTAB 16: TT$ = "CUSTOMIZE YOUR SET OF DOS CO
MMANDS!": GOSUB 3110
3080 GOSUB 3210: RETURN
3100 REM PRINT CENTER
3110 WIDTH = 20 - ( LEN (TT$) / 2): IF WIDTH < =
0 THEN PRINT TT$: RETURN
3120 HTAB WIDTH: PRINT TT$: RETURN
3200 REM CONTINUE/END
3210 VTAB 23: HTAB 12: PRINT "(ESC) TO END"
3220 VTAB 24: PRINT TAB( 8); "(SPACE) TO CONTINU
E ";
3230 PRINT "( )": HTAB 29: GET ZZ$: IF ZZ$ = CHR$
(27) OR ZZ$ = CHR$ (3) THEN TEXT : HOME : GOTO
1510
3240 IF ZZ$ = CHR$ (32) THEN RETURN
3250 CALL - 868: CALL - 1009: GOTO 3230: REM

```

```

5  REM  (C) APPLE HANDBOEK 1981
10 CALL  - 936
20 PRINT
25 VTAB 10: PRINT  SPC( 10)"*****": PRINT
   SPC( 10)"VOLGORDE": PRINT  SPC( 10)"**
   *****"
26 FOR A = 1 TO 1250: NEXT A
27 CALL  - 936
28 VTAB 3
30 PRINT "DRUK 0 VOOR EINDE PROGRAMMA"
32 PRINT : PRINT "DIT PROGRAMMA SORTEERT NA
   MEN ": PRINT "OP ALFABETISCHE VOLGORDE"

35 PRINT
40 PRINT "AANTAL TE SORTEREN GEGEVENS ?"
42 PRINT "-----"
50 VTAB 8: HTAB 32: INPUT "":B
60 IF B = 0 THEN 500
80 FOR C = 1 TO B
90 PRINT : PRINT "NAAM/EGEGEVEN ";C;" ";
100 INPUT A$(C)
110 NEXT C
120 FOR C = 1 TO B
130 FOR D = 1 TO B - 1
140 A$ = A$(D)
150 B$ = A$(D + 1)
160 IF A$ < B$ THEN 190
170 A$(D) = B$
180 A$(D + 1) = A$
190 NEXT D
200 NEXT C
210 PRINT
220 FOR C = 1 TO B
230 PRINT A$(C)
240 NEXT C
250 PRINT : INPUT "MEER GEGEVENS ? ";C$
260 IF C$ = "J" THEN 5
500 END

```

```

3  REM  APPLE HANDBOEK 1981
5  CALL  - 936
10 PRINT : PRINT  TAB( 10)"BEPALING WEEKDAG"

20 PRINT : PRINT
30 PRINT "TYPE PER VRAAG  0  VOOR EINDE"
35 PRINT
40 VTAB 10: INPUT "WELKE DAG ? ";A
42 PRINT : INPUT "WELKE MAAND ? ";B
44 PRINT : INPUT "WELK JAAR ? ";C
50 IF B <  > 0 THEN 100
70 IF A <  > 0 THEN 100
80 IF A <  > 0 THEN 100
90 GOTO 500
100 IF B > 2 THEN 130
110 B = B + 12
120 C = C - 1
130 D = A + 2 * B +  INT (.6 * (B + 1)) + C +
      INT (C / 4) -  INT (C / 100) +  INT (C /
      400) + 2
140 D =  INT ((D / 7 -  INT (D / 7)) * 7 + .5
      )
150 IF D > 0 THEN 180
160 INVERSE : PRINT : PRINT " ZATERDAG ": NORMAL

170 GOSUB 10000
175 GOTO 5
180 IF D > 1 THEN 210
190 INVERSE : PRINT : PRINT " ZONDAG "
200 GOSUB 10000: GOTO 5
210 IF D > 2 THEN 240
220 PRINT : PRINT " MAANDAG "
230 GOSUB 10000: GOTO 5
240 IF D > 3 THEN 270
250 PRINT : PRINT " DINSDAG "
260 GOSUB 10000: GOTO 5
270 IF D > 4 THEN 300
280 PRINT : PRINT " WOENSDAG "
290 GOSUB 10000: GOTO 5
300 IF D > 5 THEN 330
310 PRINT : PRINT " DONDERDAG"
320 GOSUB 10000: GOTO 5
330 PRINT : PRINT " VRIJDAG "
340 GOSUB 10000: GOTO 5
500 END
10000 FOR E = 1 TO 3500: NEXT E: NORMAL : RETURN

```



```
10 HOME
20 VTAB 3
30 D$ = CHR$ (4)
40 INPUT "AANTAL ITEMS ";I
50 FOR X = 1 TO I
60 INPUT "NAAM (MAX 10)";N$(X)
70 IF LEN (N$(X)) > 10 THEN 60
80 NEXT X
90 PRINT D$;"OPEN NAAM,L10"
95 FOR I = 1 TO X
100 PRINT D$;"WRITE NAAM,R";I
110 PRINT N$(I)
120 NEXT I
130 PRINT D$;"CLOSE NAAM"
140 HOME : VTAB 3
150 PRINT " KLAAR": FOR T = 1 TO 500: NEXT
    T
160 PRINT D$;"OPEN NAAM,L10"
165 FOR I = 1 TO X
170 PRINT D$;"READ NAAM,R";I
180 INPUT N$(I)
185 PRINT N$(I)
190 NEXT I
200 PRINT D$;"CLOSE NAAM"
210 END
```

```

1 TEXT : CALL - 936: VTAB 5: PRINT TAB( 7)"***
  ARTILLERY SIMULATOR ***": VTAB 12: PRINT "Y
  OUR MISSION IS TO BLOW UP THE OTHER ": PRINT
  "GUY BEFORE HE DOES THE SAME TO YOU.": PRINT
  : PRINT "YOU INPUT THE DATA IN THIS FORMAT:"
  : PRINT : PRINT "ANGLE TO FIRE, (COMMA) NO&
  BAGS POWDER"
2 PRINT : PRINT "EXAMPLE: 45,8.6 (CR)": PRINT
  : PRINT "WHERE (CR)= CARRIAGE RETURN": VTAB
  24: PRINT TAB( 12)"*** STAND BY ***": FOR
  A = 1 TO 3000: NEXT A: DIM A(280): B = 99: C =
  3.141592654 / 180
3 HGR : CALL - 936: B = - B: HCOLOR= 1: D = 139:
  FOR A = 1 TO 112: D = RND (1) * 10 + 140: HPLOT
  A,D TO A,159:A(A) = D: NEXT : E = INT ( RND
  (1) * 60 + 30): FOR A = 113 TO 152:F = SIN
  ((A - 112) * 4.5 + 180) * C) * E:A(A) = F +
  D: HPLOT A,F + D TO A,159: NEXT : FOR A = 15
  3 TO 279:D = RND (1) * 10 + 140: HPLOT A,D TO
  A,159:A(A) = D: NEXT : G = INT ( RND (1) * 1
  0): IF RND (1) * 100 > 50 THEN G = - G
4 HCOLOR= 3: HPLOT 112,10 TO 152,10: IF G > 0 THEN
  HPLOT 112,10 TO 125,5
5 VTAB 21: PRINT TAB( 17)"WIND:": ABS (G): IF G
  < 0 THEN HPLOT 152,10 TO 139,5
6 H = RND (1) * 100 + (( RND (1) * 10) + 5): I =
  278 - ( RND (1) * 100): HPLOT H - 3,D TO H -
  3,D - 5: HPLOT TO H + 3,D - 5: HPLOT TO H +
  3,D: HPLOT TO H - 3,D - 5: HPLOT H - 3,D TO
  H + 3,D - 5: HCOLOR= 3: HPLOT I - 3,D TO I -
  3,D - 5: HPLOT TO I + 3,D - 5: HPLOT TO I +
  3,D: HPLOT TO I - 3,D - 5: HPLOT I - 3,D TO
  I + 3,D - 5: IF B = 99 THEN 22
7 VTAB 24: INPUT "ANG,BG?": L,M:M = M * 10
8 IF B = - 99 THEN N = H
9 IF B = 99 THEN N = I
10 VTAB 21: PRINT TAB( 17)"WIND:": ABS (G): A =
  0
11 A = A + .1: O = M * A * COS (L * C): IF B = -
  99 THEN O = O + H
12 K = PEEK ( - 16336): POKE - 16336,0: IF B =
  99 THEN O = I + O
13 P = M * A * SIN (L * C) - 16 * A ^ 2: O = O -
  G * A ^ 2: IF B = 99 THEN P = M * A * ( - SIN
  (L * C)) - 16 * A ^ 2
14 HCOLOR= 0: HPLOT N,O: IF O < 3 OR O > 276 OR
  D - P < 0 THEN 21
15 IF A(O) < D - P THEN 17
16 HCOLOR= 3: HPLOT O,D - P:N = O: O = D - P: GOTO
  11
17 IF ABS (O - H) < 3 THEN 23
18 IF ABS (O - I) < 3 THEN 24
19 IF O < 3 OR O > 276 THEN 21
20 HCOLOR= 0: FOR A = 1 TO 25: R = O - 3 + RND (
  1) * 6:S = A(R) + RND (1) * 3: HPLOT R,S:K =
  PEEK ( - 16336): POKE - 16336,0: NEXT : FOR
  A = O - 3 TO O + 3:A(A) = A(A) + 3: NEXT
21 B = - B: IF B = - 99 THEN 7

```

```
22 VTAB 24: PRINT TAB( 25);: INPUT "ANG,BG?";L,  
    M:M = M * 10:L = L + 180:Q = D - 5: GOTO 8  
23 T = H: GOTO 25  
24 T = I  
25 HCOLOR= 3: FOR A = T - 10 TO T + 10 STEP 2: HPLOT  
    A,D - 10 TO T,D: FOR J = 1 TO RND (1) * 5 +  
    5:K = K + PEEK ( - 16336): NEXT : POKE - 1  
    6336,0: NEXT : FOR A = 1 TO 2500: NEXT : GOTO  
    3
```

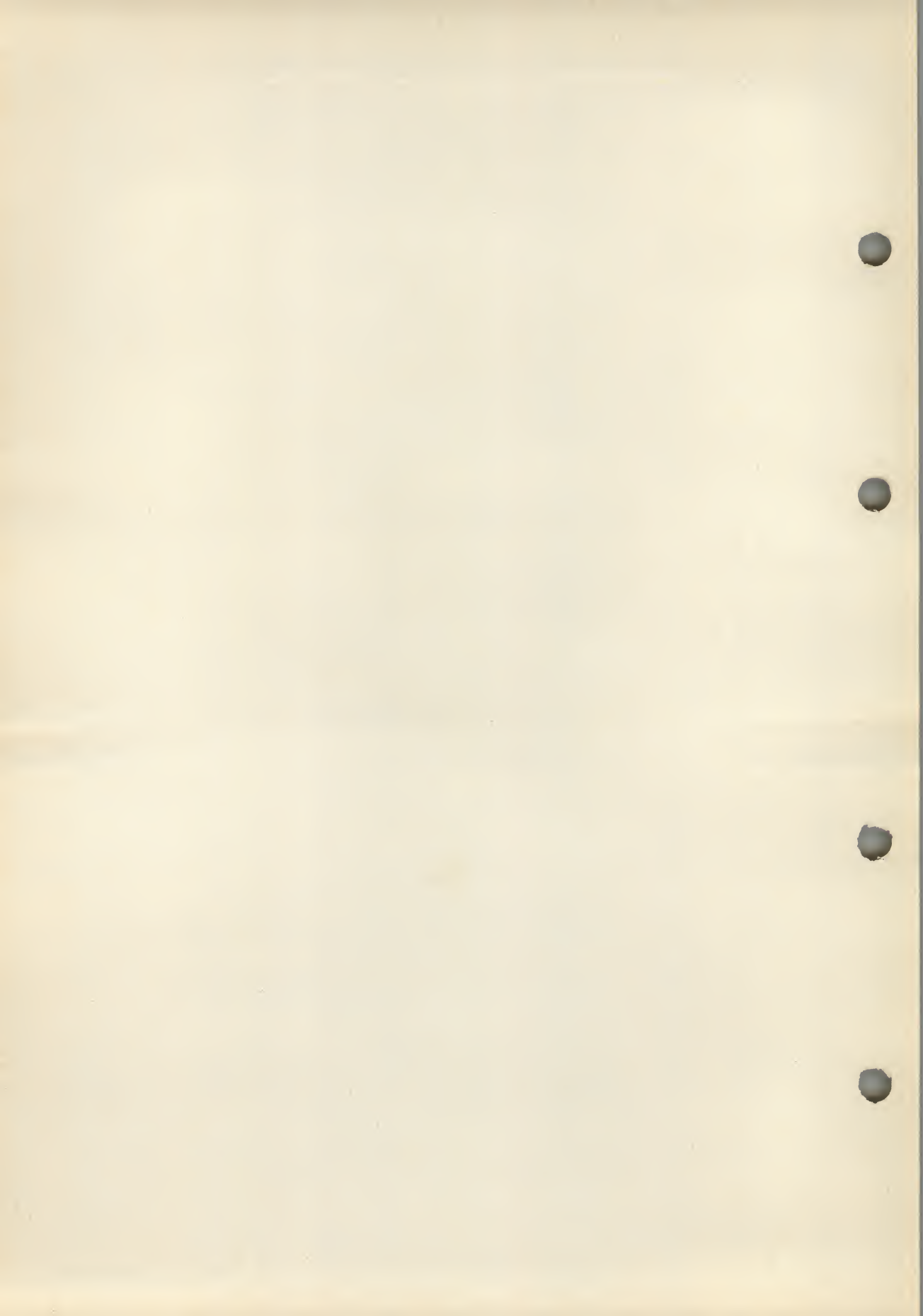


```

5 POKE 214,255
10 TEXT : HOME
11 NORMAL
30 VTAB 8: PRINT TAB( 8)"MENU": PRINT : PRINT
   TAB( 10)"1. DECIMAAL IN HEX": PRINT TAB(
   10)"2. HEX IN DECIMAAL": PRINT TAB( 10)
   "3. EINDE"
35 VTAB 17: PRINT TAB( 13)"GEEF UW KEUZE ";
   : GET A$
40 A = VAL (A$): IF A < 1 OR A > 3 THEN 10
50 ON A GOTO 100,200
60 POKE 214,0: NEW : END
100 HOME : VTAB 7: PRINT "GEEF DECIMAAL NUMM
   ER ";; INPUT NBR$
101 X = LEN (NBR$): IF X > 6 THEN 100
102 IF LEFT$ (NBR$,1) = "-" THEN 105
103 NBR = VAL (NBR$): GOTO 108
105 NBR = VAL (NBR$):NBR = NBR + 65536: GOTO
   109
108 IF NBR > 65535 THEN 100
109 GOSUB 500
110 MD256 = FN MD(NBR)
120 POKE 1,MD256: POKE 0,NBR / 256
130 POKE 60,0: POKE 61,0: POKE 62,1: POKE 63
   ,0
140 HOME : VTAB 8: PRINT TAB( 2)"DEC= ";NBR
   : " < CONTROLEER! ";; INVERSE : PRINT NBR
   $: NORMAL : CALL - 589
150 POKE 1192,160: POKE 1193,200: POKE 1194,
   197: POKE 1195,216: POKE 1196,189: POKE
   1197,160
160 GOTO 300
200 HOME : VTAB 8: INPUT "GEEF HEX NBR ";NBR
   $
205 L = LEN (NBR$): IF L = 3 THEN NBR$ = "0"
   + NBR$
206 IF L > 4 THEN 200
210 HI$ = LEFT$ (NBR$,2):LO$ = RIGHT$ (NBR$
   ,2)
220 IF LEN (NBR$) < 3 THEN HI$ = ""
230 HEX$ = "0:" + LO$ + " " + HI$ + " N D823G
   "
240 FOR I = 1 TO LEN (HEX$): POKE 511 + I, ASC
   ( MID$ (HEX$,I,1)) + 128: NEXT I: POKE 7
   2,0: CALL - 144
245 N = PEEK (0) + PEEK (1) * 256:NNB = N -
   65536
250 HOME : VTAB 9: PRINT TAB( 8)"HEX =";NBR
   $: PRINT TAB( 8)"DEC ="; PEEK (0) + PEEK
   (1) * 256;" OF: ";NNB
300 VTAB 21: INVERSE : PRINT "DRUK TOETS";: GET
   B$: POKE 0,0: POKE 1,0: GOTO 10
500 DEF FN MD(NBR) = NBR - INT (NBR / 256)
   * 256: RETURN
65535 REM (C)HANS VAN KAMPEN
   POSTBUS 2030 4200 BA GORIN
   CHEM

```

```
59998 REM APPLE HANDBOEK 1981
59999 POKE 214,255
60000 REM SUBROUTINE PRINT BEELDSCHERM
60001 TEXT : POKE 34,23: PR# 1
60002 FOR I = 0 TO 22
60003 R$ = ""
60004 FOR J = 0 TO 39
60005 R = PEEK (1024 + 40 * INT (I / 8) +
        (I - 8 * INT (I / 8)) * 128 + J)
60006 IF R < 32 THEN R = R + 64
60007 R$ = R$ + CHR$ (R)
60008 NEXT J
60010 PRINT SPC( X);R$: REM X DEFIN.
60011 NEXT I: PRINT D$;"PR#0"
60015 TEXT : HOME : POKE 214,0: NEW : END
60020 REM HVK
```

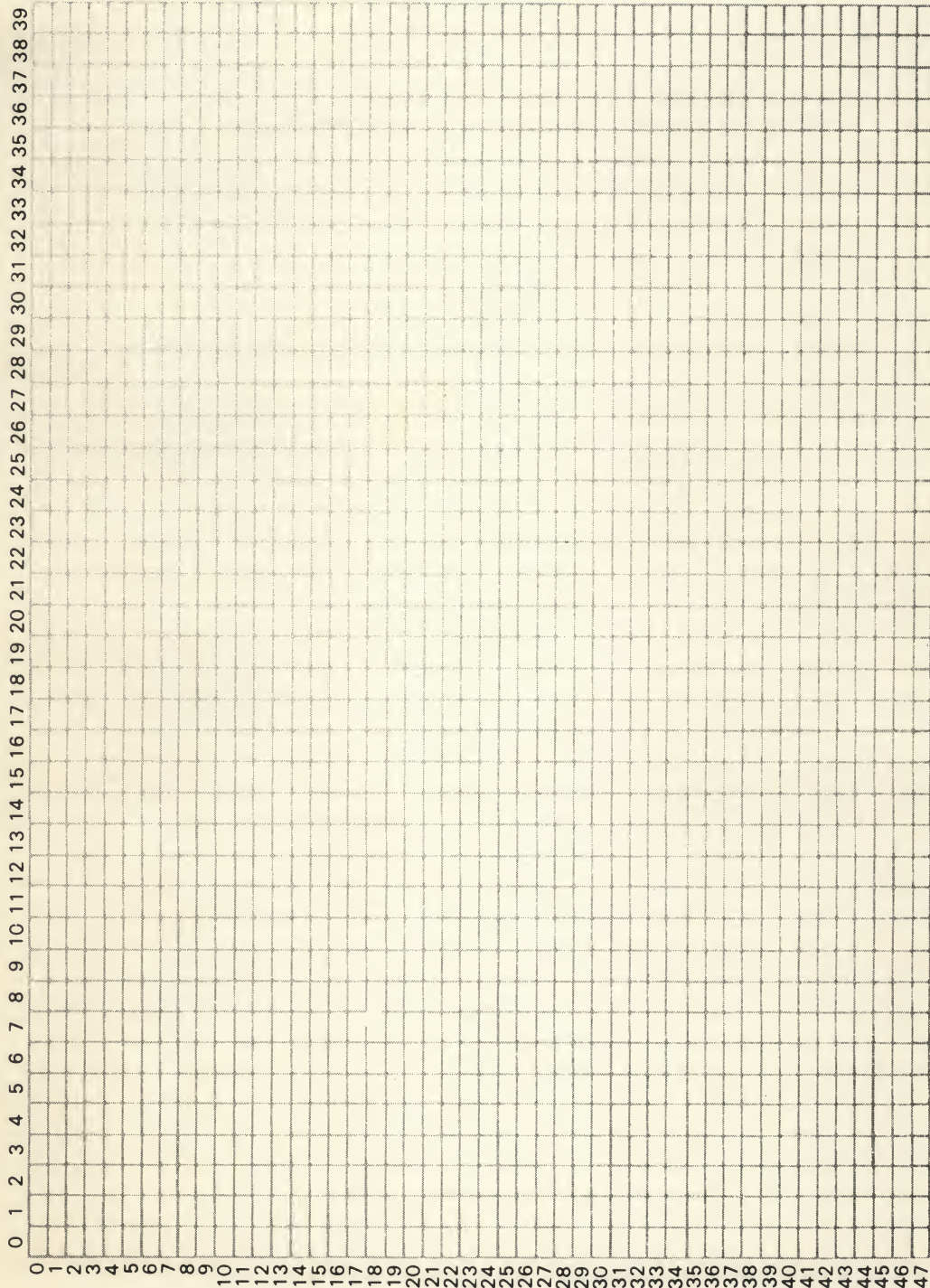







SCREEN LAYOUT FORMS

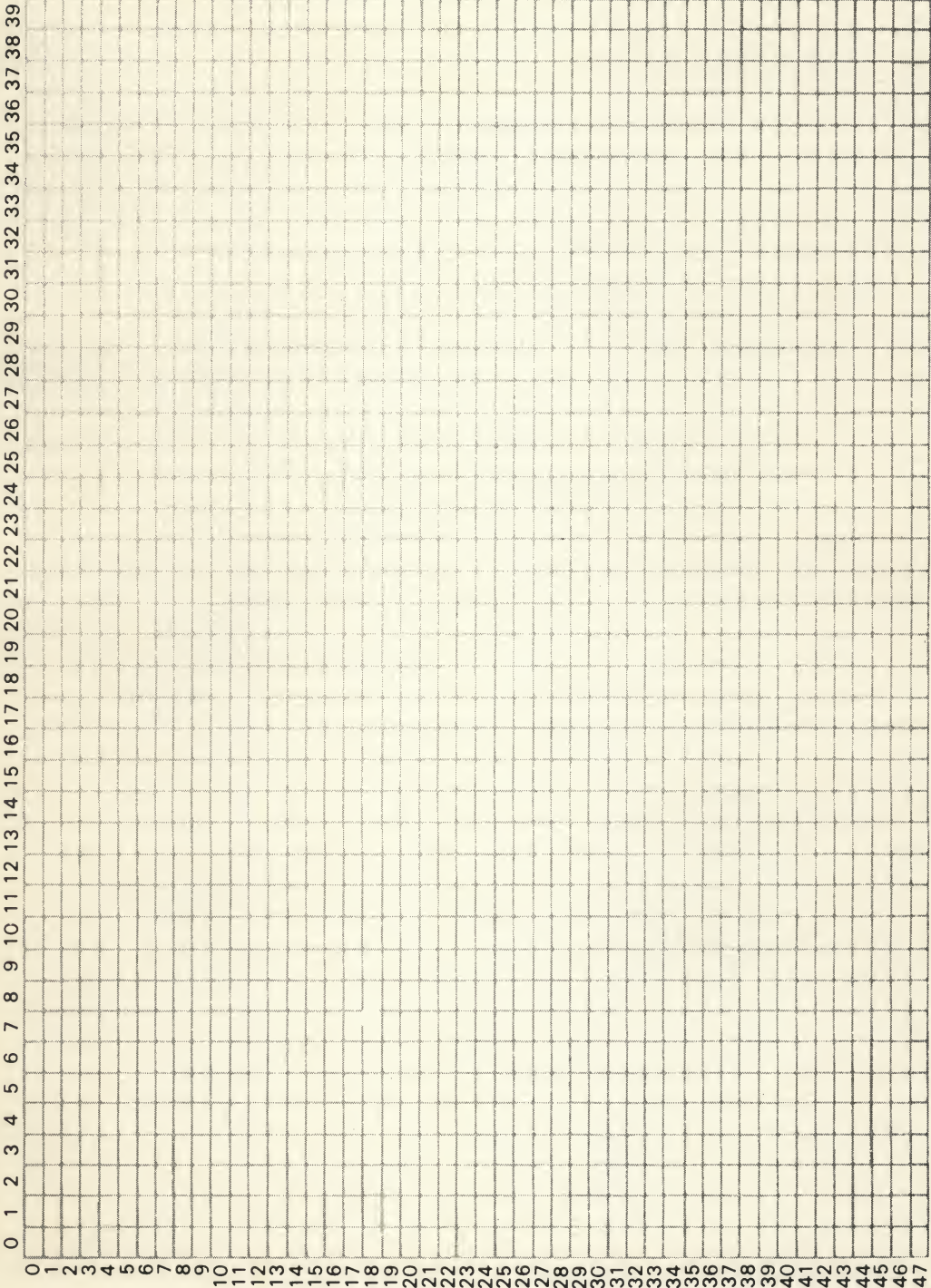
X Axis



Y Axis

SCREEN LAYOUT FORMS

X Axis

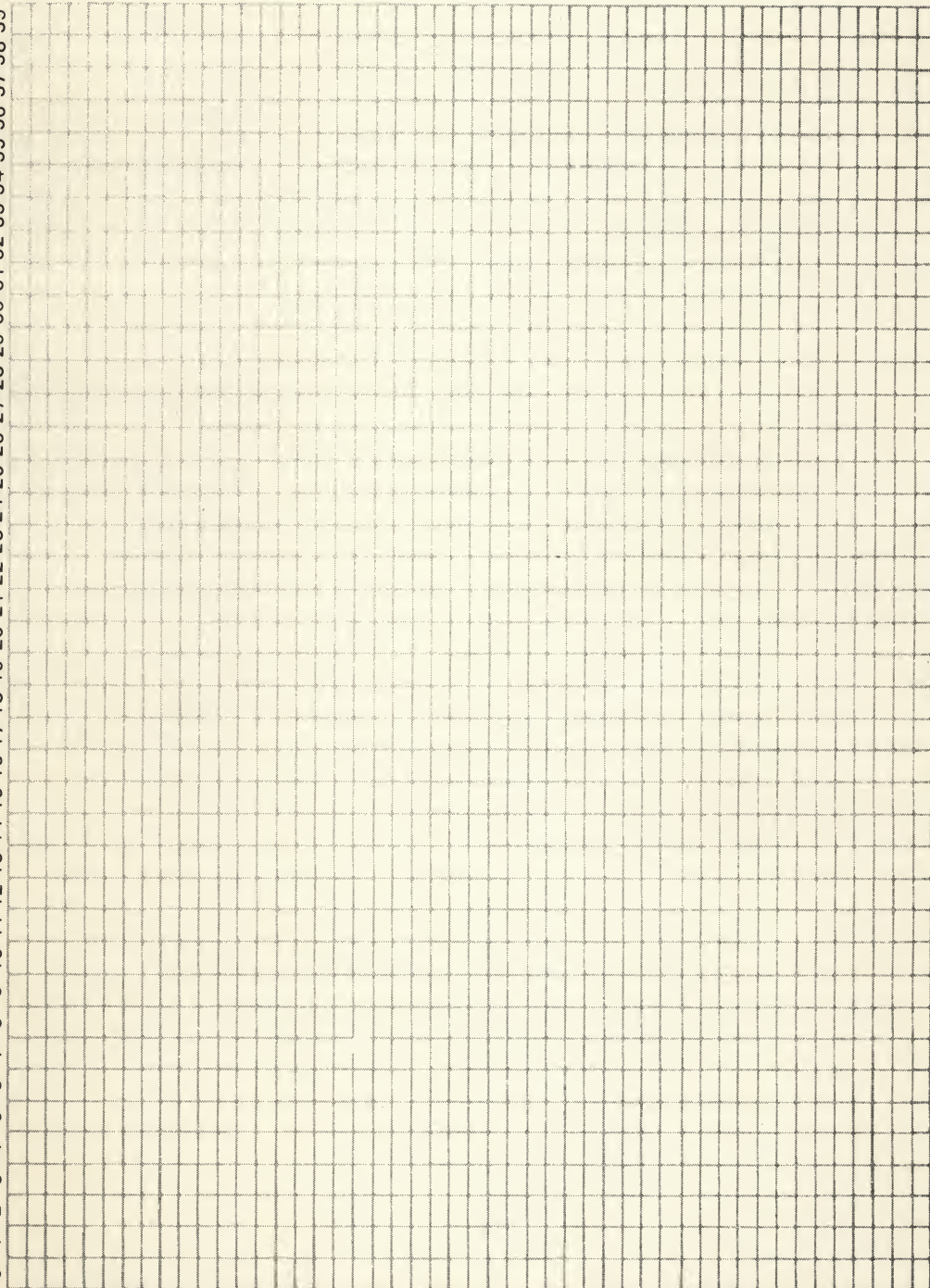


Y Axis

SCREEN LAYOUT FORMS

X Axis

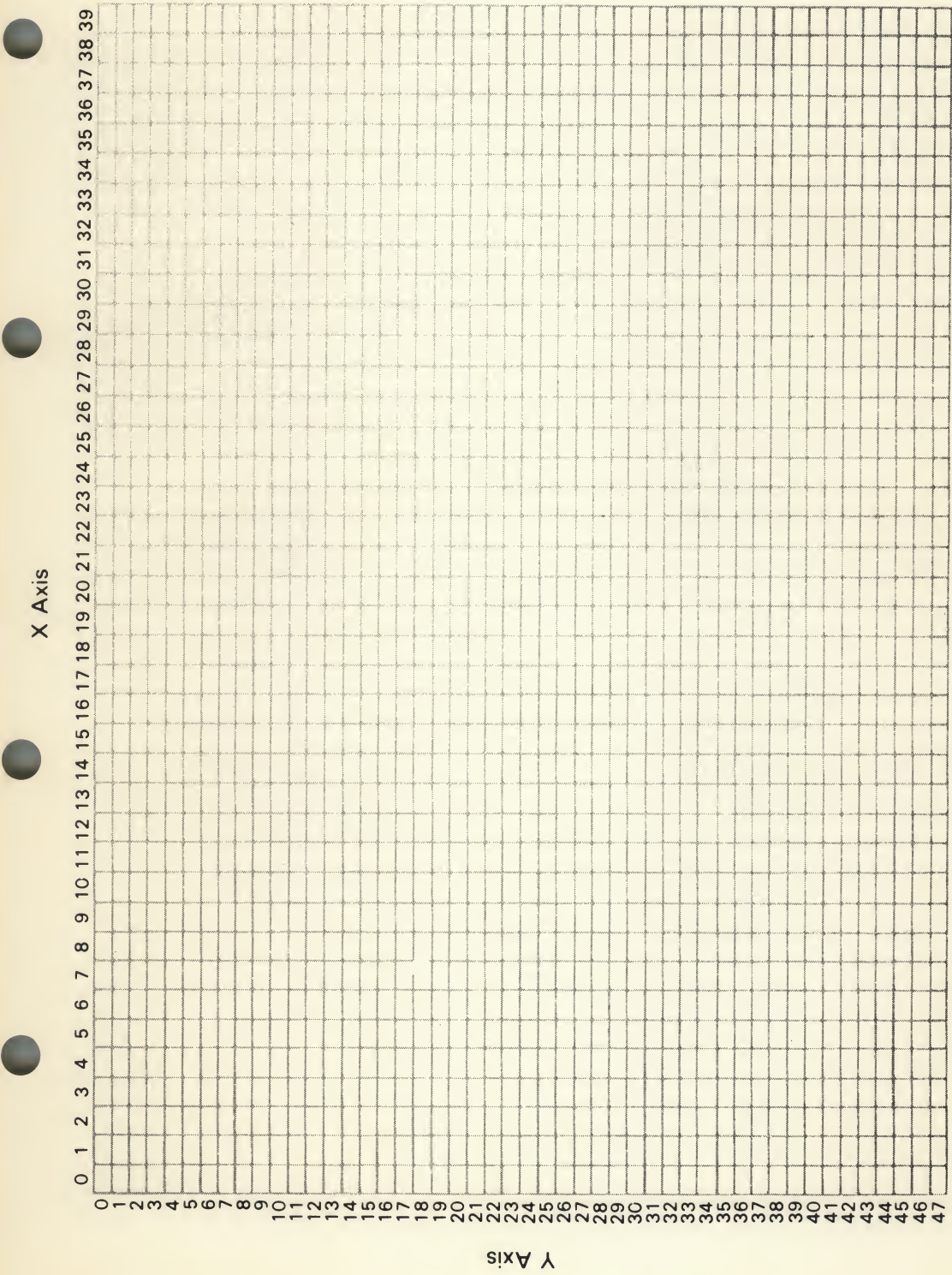
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39



Y Axis

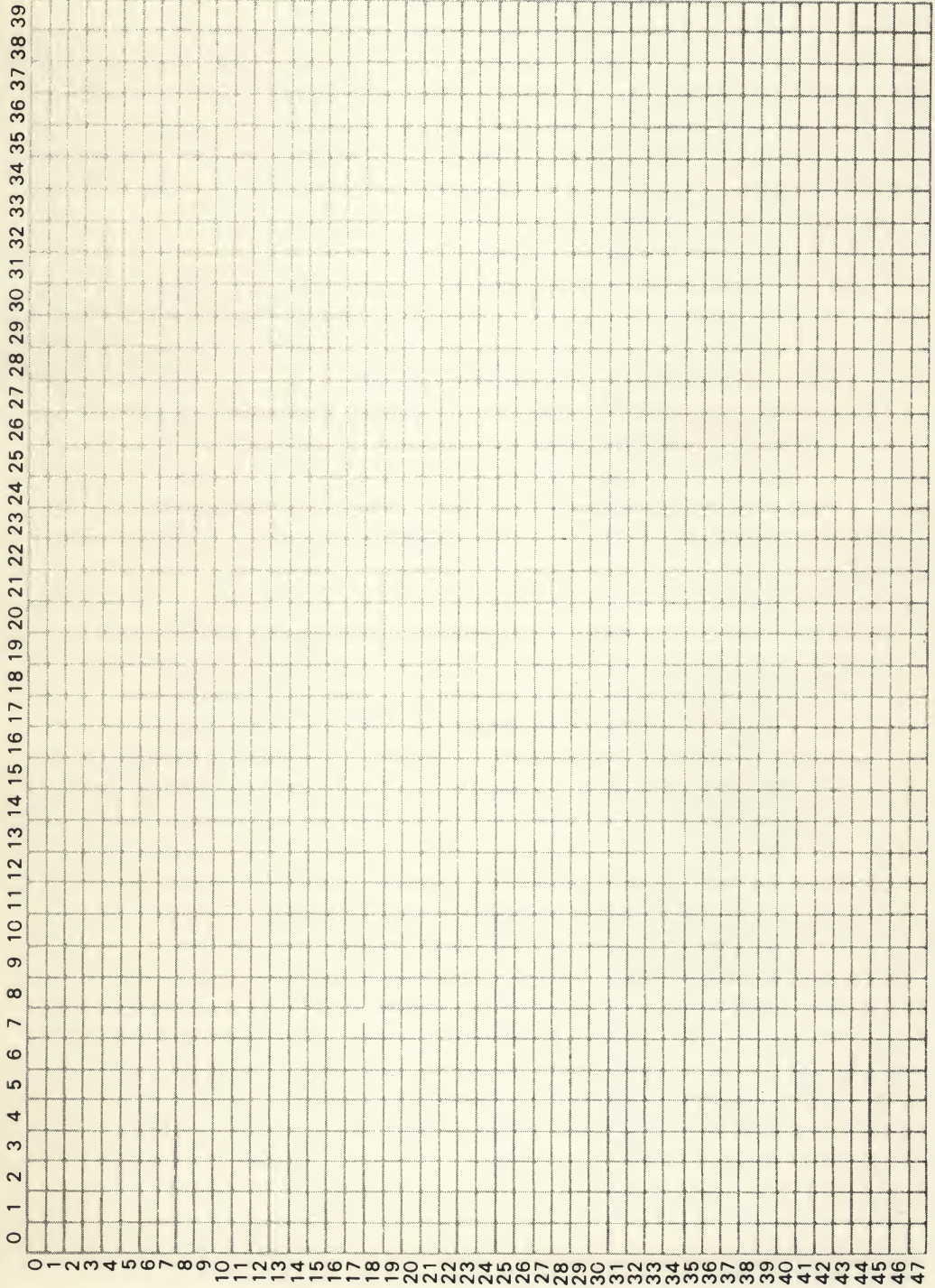
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47

SCREEN LAYOUT FORMS



SCREEN LAYOUT FORMS

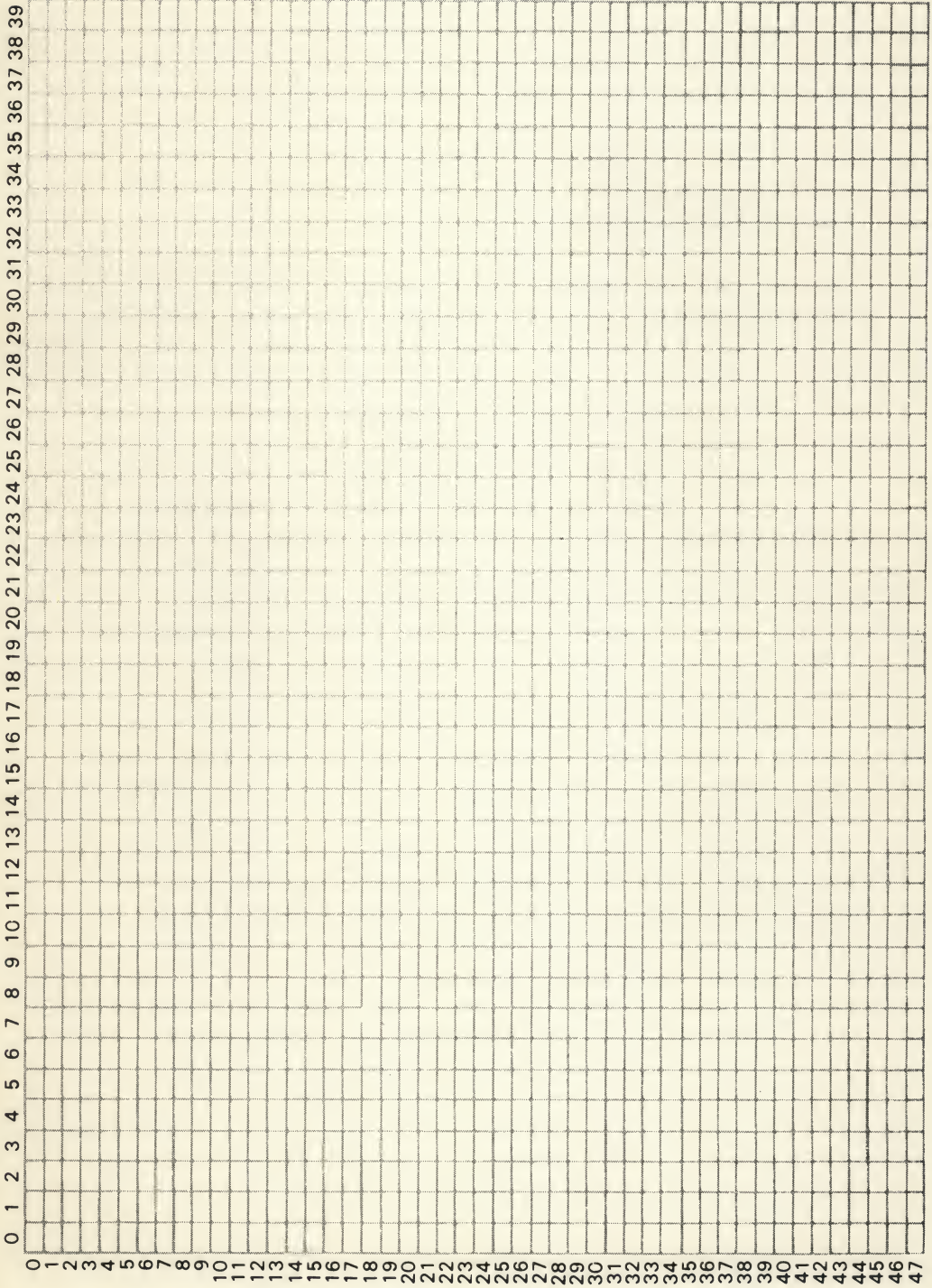
X Axis



Y Axis

SCREEN LAYOUT FORMS

X Axis



Y Axis

SCREEN LAYOUT FORMS

X Axis

Y Axis

[illegible]

SCREEN LAYOUT FORMS

X Axis

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47

Y Axis

SCREEN LAYOUT FORMS

X Axis

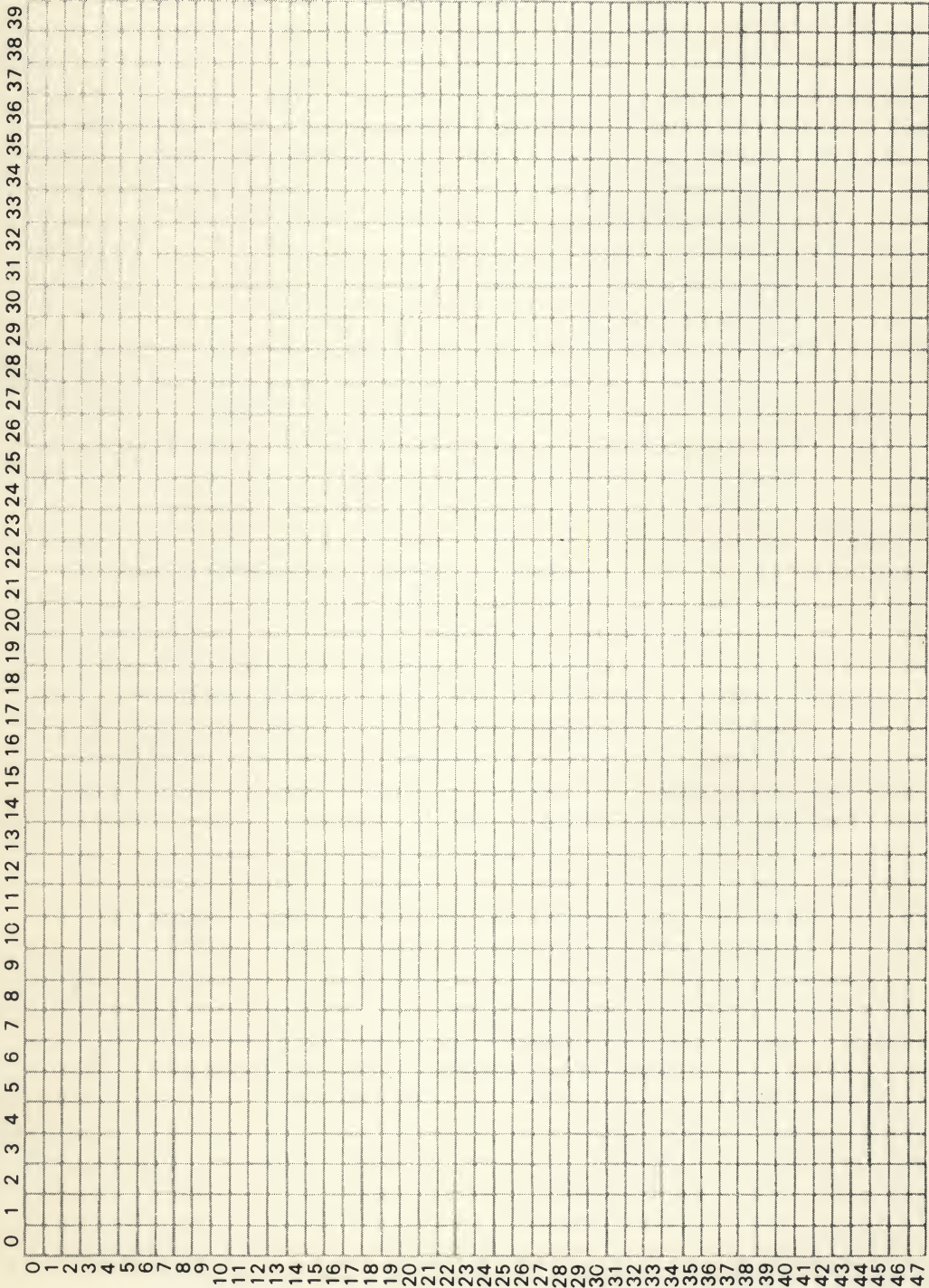
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47

Y Axis

SCREEN LAYOUT FORMS

X Axis



Y Axis

\$400 1024
 \$480 1152
 \$500 1280
 \$580 1408
 \$600 1536
 \$680 1664
 \$700 1792
 \$780 1920
 \$428 1064
 \$4A8 1192
 \$528 1320
 \$5A8 1448
 \$628 1576
 \$6A8 1704
 \$728 1832
 \$7A8 1960
 \$450 1104
 \$4D0 1232
 \$550 1360
 \$5D0 1488
 \$650 1616
 \$6D0 1744
 \$750 1872
 \$7D0 2000

0	\$00
1	\$01
2	\$02
3	\$03
4	\$04
5	\$05
6	\$06
7	\$07
8	\$08
9	\$09
10	\$0A
11	\$0B
12	\$0C
13	\$0D
14	\$0E
15	\$0F
16	\$10
17	\$11
18	\$12
19	\$13
20	\$14
21	\$15
22	\$16
23	\$17
24	\$18
25	\$19
26	\$1A
27	\$1B
28	\$1C
29	\$1D
30	\$1E
31	\$1F
32	\$20
33	\$21
34	\$22
35	\$23
36	\$24
37	\$25
38	\$26
39	\$27

Map of the Text Screen

\$400	1024
\$480	1152
\$500	1280
\$580	1408
\$600	1536
\$680	1664
\$700	1792
\$780	1920
\$428	1064
\$4A8	1192
\$528	1320
\$5A8	1448
\$628	1576
\$6A8	1704
\$728	1832
\$7A8	1960
\$450	1104
\$4D0	1232
\$550	1360
\$5D0	1488
\$650	1616
\$6D0	1744
\$750	1872
\$7D0	2000

	0	\$00
	1	\$01
	2	\$02
	3	\$03
	4	\$04
	5	\$05
	6	\$06
	7	\$07
	8	\$08
	9	\$09
	10	\$0A
	11	\$0B
	12	\$0C
	13	\$0D
	14	\$0E
	15	\$0F
	16	\$10
	17	\$11
	18	\$12
	19	\$13
	20	\$14
	21	\$15
	22	\$16
	23	\$17
	24	\$18
	25	\$19
	26	\$1A
	27	\$1B
	28	\$1C
	29	\$1D
	30	\$1E
	31	\$1F
	32	\$20
	33	\$21
	34	\$22
	35	\$23
	36	\$24
	37	\$25
	38	\$26
	39	\$27

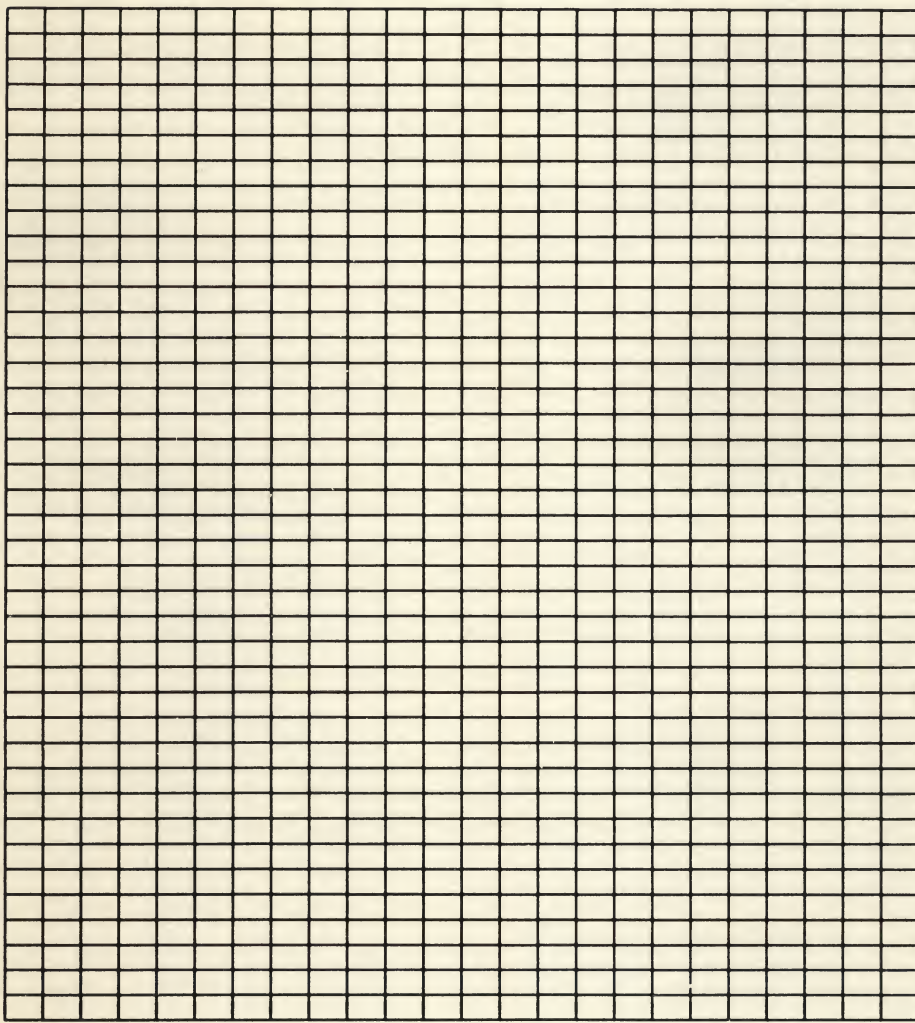
Map of the Text Screen

\$2000	8192
\$2080	8320
\$2100	8448
\$2180	8576
\$2200	8704
\$2280	8832
\$2300	8960
\$2380	9088
\$2028	8232
\$20A8	8360
\$2128	8488
\$21A8	8616
\$2228	8744
\$22A8	8872
\$2328	9000
\$23A8	9128
\$2050	8272
\$20D0	8400
\$2150	8528
\$21D0	8656
\$2250	8784
\$22D0	8912
\$2350	9040
\$23D0	9168

0	\$00
1	\$01
2	\$02
3	\$03
4	\$04
5	\$05
6	\$06
7	\$07
8	\$08
9	\$09
10	\$0A
11	\$0B
12	\$0C
13	\$0D
14	\$0E
15	\$0F
16	\$10
17	\$11
18	\$12
19	\$13
20	\$14
21	\$15
22	\$16
23	\$17
24	\$18
25	\$19
26	\$1A
27	\$1B
28	\$1C
29	\$1D
30	\$1E
31	\$1F
32	\$20
33	\$21
34	\$22
35	\$23
36	\$24
37	\$25
38	\$26
39	\$27

In each box:

\$2000 8192
 \$2080 8320
 \$2100 8448
 \$2180 8576
 \$2200 8704
 \$2280 8832
 \$2300 8960
 \$2380 9088
 \$2028 8232
 \$20A8 8360
 \$2128 8488
 \$21A8 8616
 \$2228 8744
 \$22A8 8872
 \$2328 9000
 \$23A8 9128
 \$2050 8272
 \$20D0 8400
 \$2150 8528
 \$21D0 8656
 \$2250 8784
 \$22D0 8912
 \$2350 9040
 \$23D0 9168



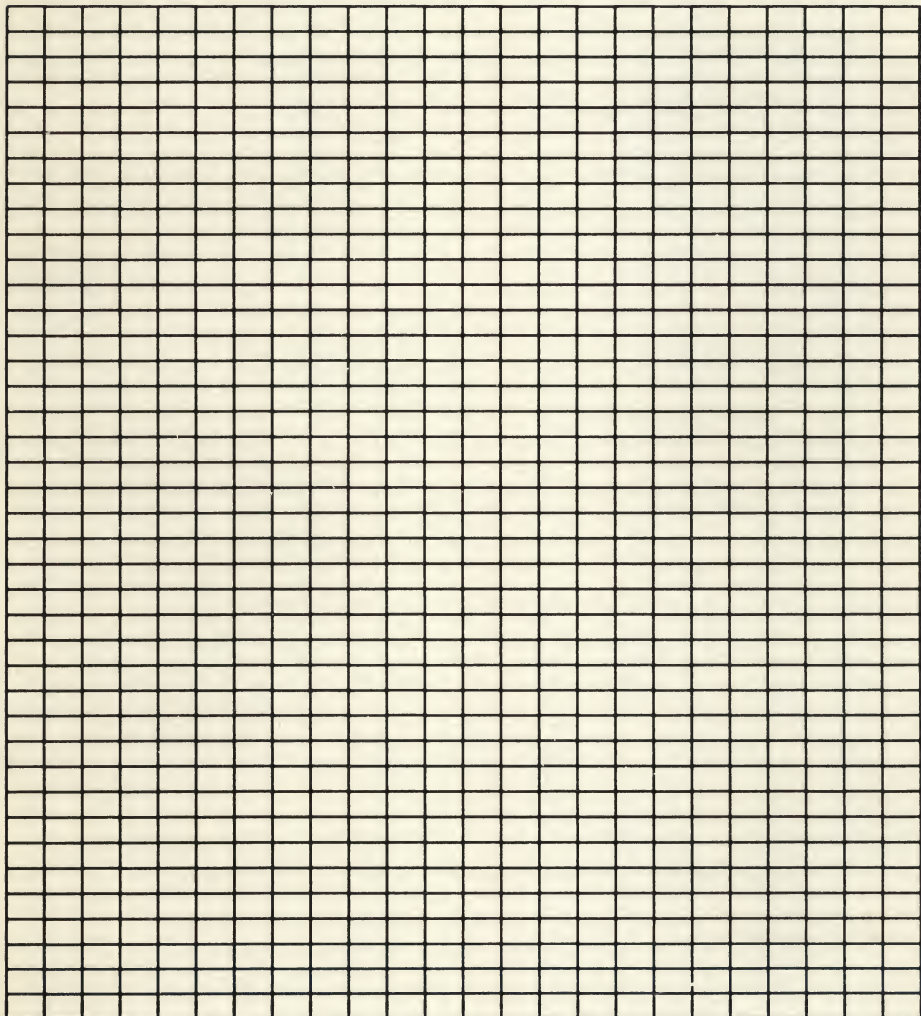
0 \$00
 1 \$01
 2 \$02
 3 \$03
 4 \$04
 5 \$05
 6 \$06
 7 \$07
 8 \$08
 9 \$09
 10 \$0A
 11 \$0B
 12 \$0C
 13 \$0D
 14 \$0E
 15 \$0F
 16 \$10
 17 \$11
 18 \$12
 19 \$13
 20 \$14
 21 \$15
 22 \$16
 23 \$17
 24 \$18
 25 \$19
 26 \$1A
 27 \$1B
 28 \$1C
 29 \$1D
 30 \$1E
 31 \$1F
 32 \$20
 33 \$21
 34 \$22
 35 \$23
 36 \$24
 37 \$25
 38 \$26
 39 \$27

Map of the High-Resolution Graphics Screen

In each box:

0 \$0000
 1024 \$0400
 2048 \$0800
 3072 \$0C00
 4096 \$1000
 5120 \$1400
 6144 \$1800
 7168 \$1C00

\$2000 8192
 \$2080 8320
 \$2100 8448
 \$2180 8576
 \$2200 8704
 \$2280 8832
 \$2300 8960
 \$2380 9088
 \$2028 8232
 \$20A8 8360
 \$2128 8488
 \$21A8 8616
 \$2228 8744
 \$22A8 8872
 \$2328 9000
 \$23A8 9128
 \$2050 8272
 \$20D0 8400
 \$2150 8528
 \$21D0 8656
 \$2250 8784
 \$22D0 8912
 \$2350 9040
 \$23D0 9168



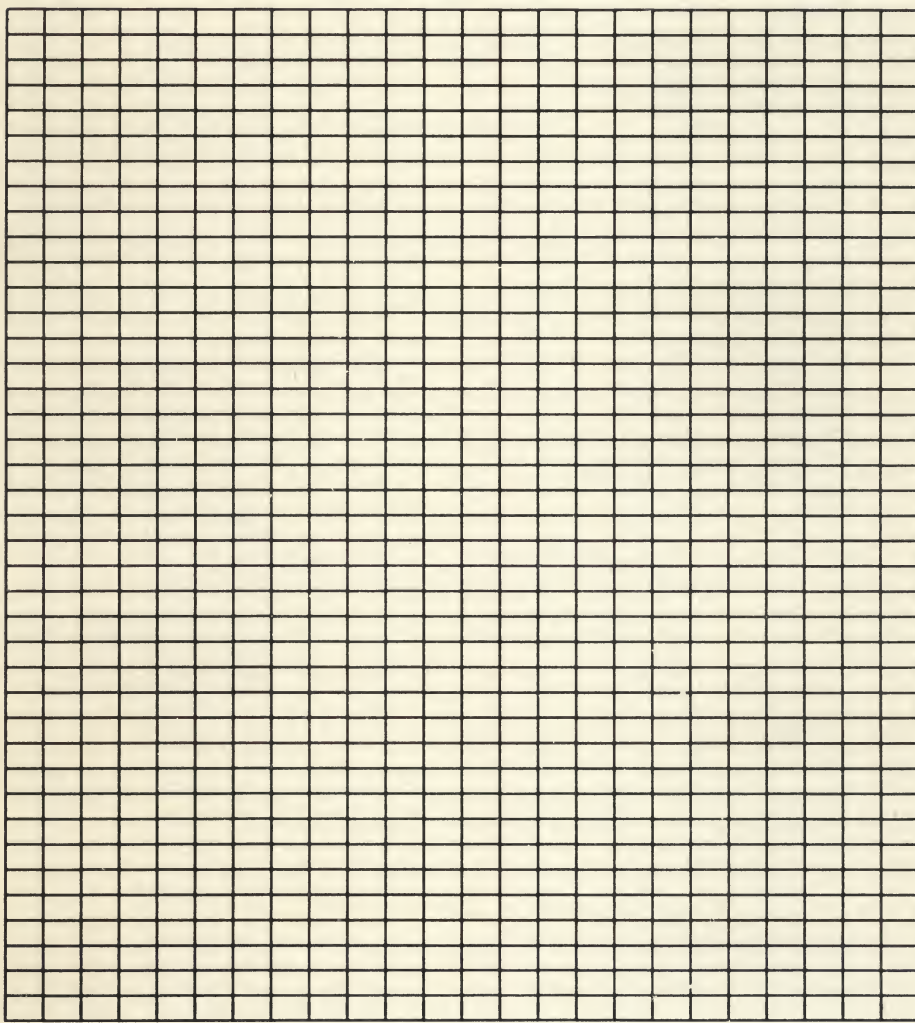
0 \$00
 1 \$01
 2 \$02
 3 \$03
 4 \$04
 5 \$05
 6 \$06
 7 \$07
 8 \$08
 9 \$09
 10 \$0A
 11 \$0B
 12 \$0C
 13 \$0D
 14 \$0E
 15 \$0F
 16 \$10
 17 \$11
 18 \$12
 19 \$13
 20 \$14
 21 \$15
 22 \$16
 23 \$17
 24 \$18
 25 \$19
 26 \$1A
 27 \$1B
 28 \$1C
 29 \$1D
 30 \$1E
 31 \$1F
 32 \$20
 33 \$21
 34 \$22
 35 \$23
 36 \$24
 37 \$25
 38 \$26
 39 \$27

Map of the High-Resolution Graphics Screen

In each box:

0 \$0000
 1024 \$0400
 2048 \$0800
 3072 \$0C00
 4096 \$1000
 5120 \$1400
 6144 \$1800
 7168 \$1C00

\$2000 8192
 \$2080 8320
 \$2100 8448
 \$2180 8576
 \$2200 8704
 \$2280 8832
 \$2300 8960
 \$2380 9088
 \$2028 8232
 \$20A8 8360
 \$2128 8488
 \$21A8 8616
 \$2228 8744
 \$22A8 8872
 \$2328 9000
 \$23A8 9128
 \$2050 8272
 \$20D0 8400
 \$2150 8528
 \$21D0 8656
 \$2250 8784
 \$22D0 8912
 \$2350 9040
 \$23D0 9168



0 \$00
 1 \$01
 2 \$02
 3 \$03
 4 \$04
 5 \$05
 6 \$06
 7 \$07
 8 \$08
 9 \$09
 10 \$0A
 11 \$0B
 12 \$0C
 13 \$0D
 14 \$0E
 15 \$0F
 16 \$10
 17 \$11
 18 \$12
 19 \$13
 20 \$14
 21 \$15
 22 \$16
 23 \$17
 24 \$18
 25 \$19
 26 \$1A
 27 \$1B
 28 \$1C
 29 \$1D
 30 \$1E
 31 \$1F
 32 \$20
 33 \$21
 34 \$22
 35 \$23
 36 \$24
 37 \$25
 38 \$26
 39 \$27

Map of the High-Resolution Graphics Screen

In each box:

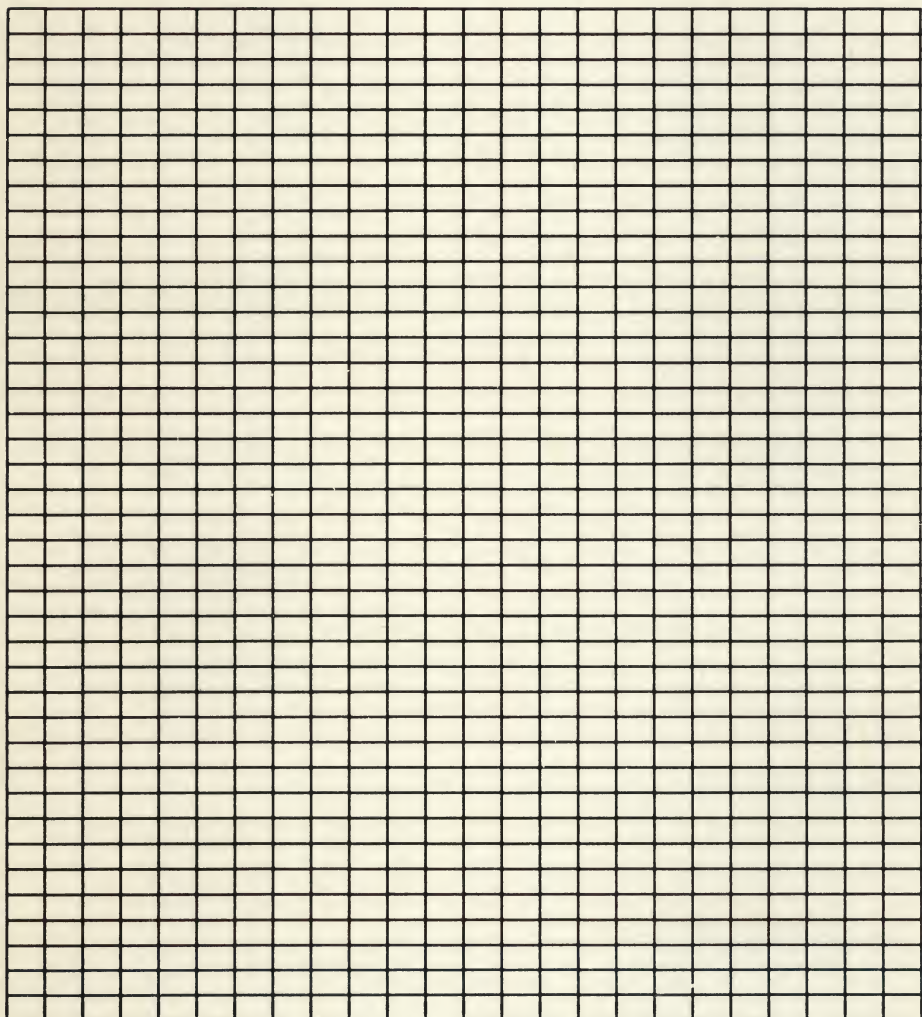
0 \$0000
 1024 \$0400
 2048 \$0800
 3072 \$0C00
 4096 \$1000
 5120 \$1400
 6144 \$1800
 7168 \$1C00

\$2000	8192
\$2080	8320
\$2100	8448
\$2180	8576
\$2200	8704
\$2280	8832
\$2300	8960
\$2380	9088
\$2028	8232
\$20A8	8360
\$2128	8488
\$21A8	8616
\$2228	8744
\$22A8	8872
\$2328	9000
\$23A8	9128
\$2050	8272
\$20D0	8400
\$2150	8528
\$21D0	8656
\$2250	8784
\$22D0	8912
\$2350	9040
\$23D0	9168

0	\$00
1	\$01
2	\$02
3	\$03
4	\$04
5	\$05
6	\$06
7	\$07
8	\$08
9	\$09
10	\$0A
11	\$0B
12	\$0C
13	\$0D
14	\$0E
15	\$0F
16	\$10
17	\$11
18	\$12
19	\$13
20	\$14
21	\$15
22	\$16
23	\$17
24	\$18
25	\$19
26	\$1A
27	\$1B
28	\$1C
29	\$1D
30	\$1E
31	\$1F
32	\$20
33	\$21
34	\$22
35	\$23
36	\$24
37	\$25
38	\$26
39	\$27

In each box:

\$2000 8192
 \$2080 8320
 \$2100 8448
 \$2180 8576
 \$2200 8704
 \$2280 8832
 \$2300 8960
 \$2380 9088
 \$2028 8232
 \$20A8 8360
 \$2128 8488
 \$21A8 8616
 \$2228 8744
 \$22A8 8872
 \$2328 9000
 \$23A8 9128
 \$2050 8272
 \$20D0 8400
 \$2150 8528
 \$21D0 8656
 \$2250 8784
 \$22D0 8912
 \$2350 9040
 \$23D0 9168



0 \$00
 1 \$01
 2 \$02
 3 \$03
 4 \$04
 5 \$05
 6 \$06
 7 \$07
 8 \$08
 9 \$09
 10 \$0A
 11 \$0B
 12 \$0C
 13 \$0D
 14 \$0E
 15 \$0F
 16 \$10
 17 \$11
 18 \$12
 19 \$13
 20 \$14
 21 \$15
 22 \$16
 23 \$17
 24 \$18
 25 \$19
 26 \$1A
 27 \$1B
 28 \$1C
 29 \$1D
 30 \$1E
 31 \$1F
 32 \$20
 33 \$21
 34 \$22
 35 \$23
 36 \$24
 37 \$25
 38 \$26
 39 \$27

Map of the High-Resolution Graphics Screen

In each box:

0 \$0000
 1024 \$0400
 2048 \$0800
 3072 \$0C00
 4096 \$1000
 5120 \$1400
 6144 \$1800
 7168 \$1C00

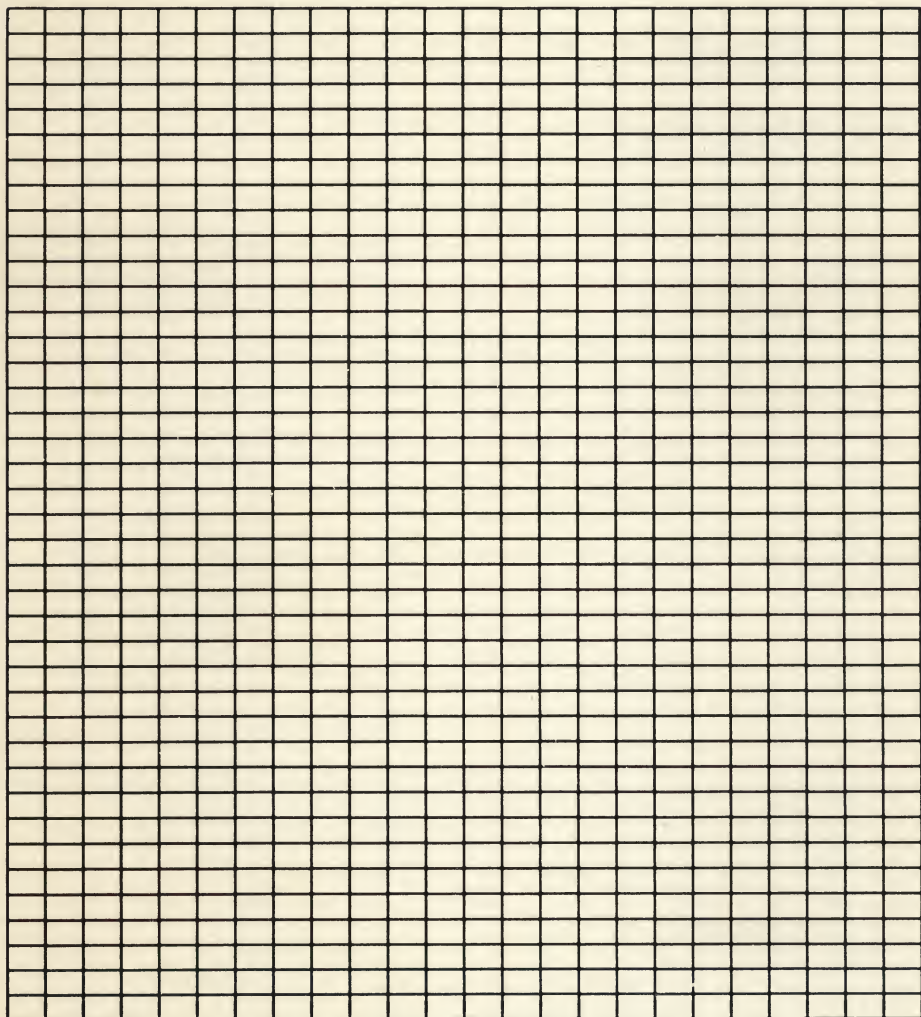
\$2000	8192
\$2080	8320
\$2100	8448
\$2180	8576
\$2200	8704
\$2280	8832
\$2300	8960
\$2380	9088
\$2028	8232
\$20A8	8360
\$2128	8488
\$21A8	8616
\$2228	8744
\$22A8	8872
\$2328	9000
\$23A8	9128
\$2050	8272
\$20D0	8400
\$2150	8528
\$21D0	8656
\$2250	8784
\$22D0	8912
\$2350	9040
\$23D0	9168

0	\$00
1	\$01
2	\$02
3	\$03
4	\$04
5	\$05
6	\$06
7	\$07
8	\$08
9	\$09
10	\$0A
11	\$0B
12	\$0C
13	\$0D
14	\$0E
15	\$0F
16	\$10
17	\$11
18	\$12
19	\$13
20	\$14
21	\$15
22	\$16
23	\$17
24	\$18
25	\$19
26	\$1A
27	\$1B
28	\$1C
29	\$1D
30	\$1E
31	\$1F
32	\$20
33	\$21
34	\$22
35	\$23
36	\$24
37	\$25
38	\$26
39	\$27

In each box:

0	\$0000
1024	\$0400
2048	\$0800
3072	\$0C00
4096	\$1000
5120	\$1400
6144	\$1800
7168	\$1C00

\$2000 8192
 \$2080 8320
 \$2100 8448
 \$2180 8576
 \$2200 8704
 \$2280 8832
 \$2300 8960
 \$2380 9088
 \$2028 8232
 \$20A8 8360
 \$2128 8488
 \$21A8 8616
 \$2228 8744
 \$22A8 8872
 \$2328 9000
 \$23A8 9128
 \$2050 8272
 \$20D0 8400
 \$2150 8528
 \$21D0 8656
 \$2250 8784
 \$22D0 8912
 \$2350 9040
 \$23D0 9168



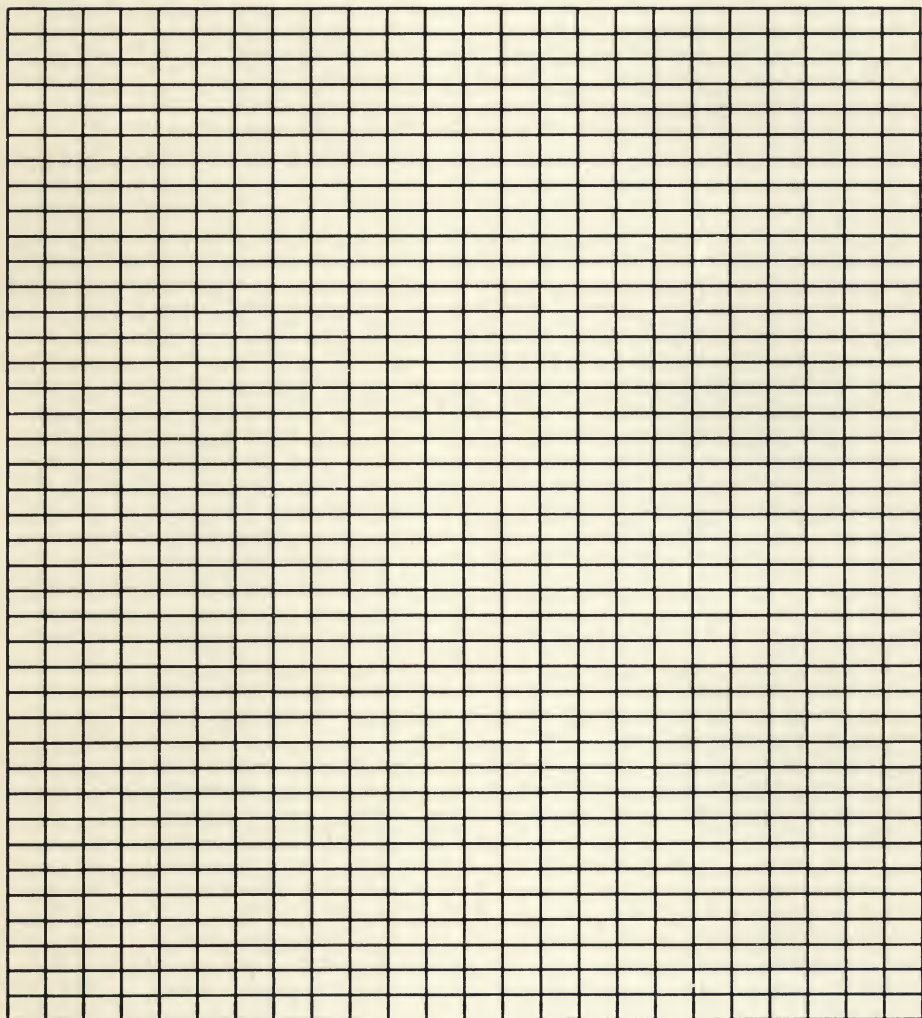
0 \$00
 1 \$01
 2 \$02
 3 \$03
 4 \$04
 5 \$05
 6 \$06
 7 \$07
 8 \$08
 9 \$09
 10 \$0A
 11 \$0B
 12 \$0C
 13 \$0D
 14 \$0E
 15 \$0F
 16 \$10
 17 \$11
 18 \$12
 19 \$13
 20 \$14
 21 \$15
 22 \$16
 23 \$17
 24 \$18
 25 \$19
 26 \$1A
 27 \$1B
 28 \$1C
 29 \$1D
 30 \$1E
 31 \$1F
 32 \$20
 33 \$21
 34 \$22
 35 \$23
 36 \$24
 37 \$25
 38 \$26
 39 \$27

Map of the High-Resolution Graphics Screen

In each box:

0 \$0000
 1024 \$0400
 2048 \$0800
 3072 \$0C00
 4096 \$1000
 5120 \$1400
 6144 \$1800
 7168 \$1C00

\$2000 8192
 \$2080 8320
 \$2100 8448
 \$2180 8576
 \$2200 8704
 \$2280 8832
 \$2300 8960
 \$2380 9088
 \$2028 8232
 \$20A8 8360
 \$2128 8488
 \$21A8 8616
 \$2228 8744
 \$22A8 8872
 \$2328 9000
 \$23A8 9128
 \$2050 8272
 \$2150 8528
 \$21D0 8656
 \$2250 8784
 \$22D0 8912
 \$2350 9040
 \$23D0 9168



0 \$00
 1 \$01
 2 \$02
 3 \$03
 4 \$04
 5 \$05
 6 \$06
 7 \$07
 8 \$08
 9 \$09
 10 \$0A
 11 \$0B
 12 \$0C
 13 \$0D
 14 \$0E
 15 \$0F
 16 \$10
 17 \$11
 18 \$12
 19 \$13
 20 \$14
 21 \$15
 22 \$16
 23 \$17
 24 \$18
 25 \$19
 26 \$1A
 27 \$1B
 28 \$1C
 29 \$1D
 30 \$1E
 31 \$1F
 32 \$20
 33 \$21
 34 \$22
 35 \$23
 36 \$24
 37 \$25
 38 \$26
 39 \$27

Map of the High-Resolution Graphics Screen

In each box:

0 \$0000
 1024 \$0400
 2048 \$0800
 3072 \$0C00
 4096 \$1000
 5120 \$1400
 6144 \$1800
 7168 \$1C00





AANTEKENINGEN

AANTEKENINGEN

AANTEKENINGEN

AANTEKENINGEN

AANTEKENINGEN

AANTEKENINGEN

AANTEKENINGEN

AANTEKENINGEN

AANTEKENINGEN

AANTEKENINGEN

AANTEKENINGEN

AANTEKENINGEN

AANTEKENINGEN

AANTEKENINGEN

AANTEKENINGEN

AANTEKENINGEN

AANTEKENINGEN

AANTEKENINGEN

AANTEKENINGEN

AANTEKENINGEN
